The Dissertation Committee for Mei Yang Certifies that this is the approved version of the following dissertation:

Using Advanced Tabu Search Techniques to Solve Airline Disruption Management Problems

Committee:

J. Wesley Barnes, Supervisor

T. Glenn Bailey

Julian Pachon

David Morton

John Hasenbein

Erhan Kutanoglu

Using Advanced Tabu Search Techniques to Solve Airline Disruption Management Problems

by

Mei Yang, B.E.; M.E.; M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2007

Acknowledgements

I would like to express my sincerest gratitude to my supervisor, Dr. J. Wesley Barnes, who guided me through my graduate study. His insightful guidance and unwavering support made this dissertation possible. I am also indebted to his continuous encouragement and inspiration.

I would like to thank all the other members of my dissertation committee, Dr. T. Glenn Bailey, Dr. Julian Pachon, Dr. David Morton, Dr. John Hasenbein and Dr. Erhan Kutanoglu, for their invaluable comments and suggestions.

My thanks also go to Dr. Melba M. Crawford, for her support and guidance in the early stages of my doctoral study. I would also like to thank Michael Argüello and Benjamin Thengvall for providing information about their earlier research work.

I would like to thank my friends for offering me suggestions with regard to doctoral study based on their own experience. They are: Keith Hutchison, Yang-chi Chen, Dongmei Zhou and Jianhua Ren.

I must thank my family for their encouragement and support. My parents-in-law helped with childcare and housework. My husband Wanwan encouraged me throughout my study. I highly appreciate what they have done for me in the past few years. And to my son Kevin, thank you for bringing much joy to me.

This research has been supported by the Air Force Office of Scientific Research.

Using Advanced Tabu Search Techniques to Solve Airline Disruption Management Problems

Publication No._____

Mei Yang, Ph.D.

The University of Texas at Austin, 2007

Supervisor: J. Wesley Barnes

Disruption Management in the airline industry plays an important role in airline operations. The goal of disruption management is to minimize the costs associated with disruptions while returning to the original schedule. Methodologies using advanced tabu search (TS) were investigated to solve two flight rescheduling problems: the aircraft grounding problem and the reduced station capacity problem. The objectives of both problems were to minimize the schedule recovery costs associated with flight schedule modifications and deviations from the original route, which are composed of the sum of delay costs, cancellation costs and aircraft route swap costs. Reflecting the cost of the deviation from the original route, the swap cost was modeled as a non-linear function of the swaps of aircraft between routes. In each problem, a stand-alone tabu search approach was constructed to holistically minimize the sum of the cost of delays, cancellations and swaps. Next a hybrid method which combined a time-space network flow model with side constraints and a limited tabu search was created which attacked the problem in two steps: first, the total cost of delays and cancellations was minimized by the network flow

model; second, a limited tabu search was conducted to minimize the number of swaps. A second hybrid method was then developed, which utilized the result from the first hybrid method as starting solution for the stand-alone tabu search. The results of the experiments performed with the hybrid methods clearly indicate that integrating TS with classical optimization methods has marked potential for improving the results of a disruption management technique.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Disruption management has been attracting more and more research interest recently. Yu and Qi (2004) formally stated it as follows: "At the beginning of a business cycle, an optimal or near optimal operational plan is obtained by using certain optimization models and solution schemes. When such an operational plan is executed, disruptions may occur from time to time caused by internal and external uncertain factors. As the result, the original operational plan may not remain optimal, or even feasible. Consequently, we need to dynamically revise the original plan and obtain a new one that reflects the constraints and objectives of the evolved environment while minimizing the negative impact of disruption." Recent research on disruption management has addressed applications in private sector airline fleets, supply chains, machine scheduling and project management. An extensive literature review on these problems covered by recent research can be found in Yu and Qi (2004).

The airline industry deals with disruption management as one of the critical daily tasks in the operations. Storms, mechanical failures, runway problems and many other unforeseen scenarios may lead to unexpected resource shortages including aircraft, airport gates or station capacity, which may cause the original plan to be disrupted. In addition, correcting the cause of the disruption is usually insufficient for a return to the original operational plan. For example, only correcting an aircraft's mechanical failure does not assure that the correct number of aircraft are available at each airport (i.e., *station*) to resume the original flight schedule on the next day. This is an important consideration because the original schedule is *highly* optimized and any deviation from

the original schedule can incur large costs for the airline. Furthermore, disruption management decisions must be made in a timely fashion. It is very important to be able to return to and resume the original schedule as soon as possible.

Ineffective and inefficient management of schedule disruptions can cause tremendous costs to an airline company. A recent example of such mismanagement was exemplified by JetBlue Airways' operational difficulties in February 2007 caused by an ice storm hitting the Eastern United States. JetBlue's insufficient disruption management resulted in $41 million of costs associated with such things as passenger refunds, travel vouchers for future bookings and incremental costs associated with such things as hiring overtime crews.

Each airline has an operation control center (OCC) to deal with day-to-day disruption management problems. Many airlines heavily depend only on the OCC staff's experience and intuition to decide what to do. Unfortunately, because of the massive information that must be considered (involving such things as published schedules, aircraft, stations, crews), it is virtually impossible to make effective decisions in such a time-critical situation.

A frequent airline disruption is the unavailability of one or more aircraft. Aircraft are expensive to purchase, operate and maintain and comprise one of the largest components contributing to the total airline operating cost. For this reason, it is very rare to have surplus aircraft at any station. Aircraft can become temporarily unavailable due to many causes including mechanical failure, delayed arrival or the lack of available pilots or crew. In a published original schedule each utilized aircraft is assigned an ordered sequence of flights, a *route*. Ferry flights, with no passengers, to restore the original schedule's *station balance*, i.e., the required number and type of aircraft at each station to

resume the original schedule, is very rarely employed due to the associated large cost of such an action.

Another challenging disruption is that of a reduced station airplane capacity. Such things as gate unavailability or inclement weather can cause the maximum allowed number of aircraft in the station reduced to a lower capacity than usual. Most major passenger airlines based in the United States use a "hub-and-spoke" network to route their planes. *Hubs* are a set of large central airports that most aircraft routes pass through. *Spokes* are comprised of smaller airports serviced by routes departing from and returning to the hubs. The hub-and-spoke system saves money and provides better passenger service. Since a hub serves as a transportation "center," a reduction of hub capacity has a much greater affect on the published schedule than capacity reduction at a spoke station. Suppose that a hub experiences disruption like a winter ice storm. This would greatly reduce the hub capacity by making it possible to service only a small proportion of the usual number of aircraft. This could cause some flights to be cancelled and could also cause many aircraft on the ground destined for the disrupted hub to experience significant delays. In some scenarios, aircraft already in the air may have to divert to alternate airports. In the extreme case where a serious tornado is known to be nearing the hub, all aircraft would be required to exit the hub and not return until the danger had passed and flight operations could be resumed**.**

In problems like these, the disruption management plan must: (1) minimize the cost associated with the schedule revision, flight cancellations and delays, (2) minimize the route deviations from the original schedule; and (3) resume the original schedule by the end of recovery window. Since there are multiple criteria associated with these problems, there is not always a dominant solution but rather a set of competing solutions. In this case, a decision maker must select a solution from that set based upon experience

and preferences that are not necessarily explicitly included in the model. A method that quickly provides an ensemble of "good" solutions of diverse character constitutes a *decision aid* that can be used to great advantage in many scenarios.

This dissertation documents the research efforts that led to the formulation and implementation of effective and efficient solution tabu search based methodologies for the two airline disruption management problems described above.

Chapter 2

Background and Related Work

This chapter provides brief descriptions of the airline scheduling process and airline disruption management. This is followed by a brief literature review of research associated with flight rescheduling, an overview of the tabu search methodology and a selected literature review of published applications of tabu search.

2.1 AIRLINE SCHEDULING

The daily flight schedule of an airline is generated to maximize profit while satisfying many operational constraints such as aircraft capacity, passenger volume, government regulations, union agreements, crew availability, aircraft maintenance requirement, and the airport runway/gate schedule. The process of generating the daily schedule usually is composed of three steps: (1) design the flight network; (2) solve the fleet assignment problem; and (3) determine the aircraft routes.

2.2 AIRLINE DISRUPTION MANAGEMENT

Various things, including aircraft mechanical failure, crew unavailability, severe weather, may disrupt the flight schedule. Disruptions require revisions to the original schedule followed by recovery to the published schedule within a specified time horizon. The goals of this process are to minimize the costs of the disruption, including lost revenue, crew cost, and customer good will, and to resume the original schedule in a timely fashion.

Research in airline disruption management problems in the literature addressed problems like aircraft shortage (Jarrah et al. 1993, Argüello et al. 1997, Thengvall et al.

2000, Bard et al. 2001, Rosenberger et al. 2003), airport closure (Yan and Lin 1997, Thengvall et al. 2001, Rosenberger et al. 2003), ground holding, i.e., delaying aircraft departures, (Vranas et al. 1994, Luo et al. 1997), crew rescheduling (Wei et al. 1997, Lettovský et al. 2000, Yu et al. 2003). The models and methods applied include network flow models (Jarrah et al. 1993, Yan and Lin 1997, Thengvall et al. 2001), integer programming formulations (Vranas et al. 1994, Luo et al. 1997, Wei et al. 1997, Lettovský et al. 2000, Yu et al. 2003), set packing models (Rosenberger et al. 2003), and heuristics (Argüello et al. 1997, Wei et al. 1997).

Literature reviews of previous research on airline disruption management problems can be found in Filar et al. (2001), Kohl et al. (2004), and Yu and Qi (2004). In this research, we restrict our attention to flight rescheduling problems.

2.3 FLIGHT RESCHEDULING

Argüello et al. (1997) and Argüello (1997) present a time-band model and a greedy randomized adaptive search procedure (GRASP) to reconstruct aircraft routings in response to groundings and delays. The GRASP generates a solution neighborhood about the incumbent solution, stores a subset of the most desirable solutions, and then randomly selects one of them to be the new incumbent solution. The objective is to minimize the total cost for delays and cancellations. The deviation from the original route is not considered.

Bard, Yu and Argüello (2001) modeled the same problem with an integer network flow model with side constraints. In the model, flight arcs are placed in the network to allow all feasible flight connections in aircraft route. The transformation procedure is polynomial with respect to the number of aircraft and flights in the schedule. A relaxed linear model is first solved. If a fractional solution is obtained then the mixed integer program solver is called to obtain an integer solution. The quality of the solution can be

improved by decreasing the time-band length. Their algorithm was implemented using CPLEX on data provided by Continental Airlines.

Thengvall et al. (2000) solved the aircraft grounding problem using a time-space network flow model with side constraints. The delays are incorporated by adding a series of arcs with a delay cost for each flight. Deviations in aircraft routing were discouraged by protection arcs which encouraged preservation of the original routes. The solution varies depending on the incentives provided by the protection arcs, the costs of delaying flights and the number of delay options. The use of protection arcs does not precisely conform to the current commercial airline management practice for minimizing the number of aircraft route swaps when compared with the original schedule (Pachon 2007a).

Thengvall et al. (2001) investigated three network-type models to determine a recovery schedule for aircraft following a hub closure. A space-time network representation was used to model the problem and maximize the total profit associated with the recovery period schedule. A set of representative problems were solved using CPLEX's MIP solver which first implements a barrier algorithm and then switches to the dual simplex algorithm for the final implicit enumeration using a branch and bound methodology.

Loeve et al. (2001a, 2002) proposed two heuristics modifying the assignment of aircrafts to flights by means of swaps. One is an iterated local search with a variable neighborhood search; the other is the steepest ascent local search. The objective is to maximize revenue minus delay cost and cancellation cost.

Rosenberger et al. (2003) modeled the flight rescheduling problem as a set-packing problem where possible new routes for each aircraft were generated a priori and then the optimal set of routes was determined. This model was computationally intensive

since for each aircraft all possible routes must be generated. An aircraft selection heuristic was introduced to select a subset of aircraft for optimization prior to generating new routes.

Andersson (2001, 2006) are the only previous publications describing research using tabu search (TS) for an aircraft grounding disruption management problem. This *primitive* TS method with static tabu tenure approaches the aircraft grounding problem utilizing a strictly linear objective function that incorporates linear coefficients to attempt to implicitly capture the costs of flight cancellation and swaps of aircraft between routes. These coefficients must be determined by the user of the algorithm for any change of aircraft assigned to a specific flight. In addition, a simple linear function, comprised of a coefficient multiplying the flight delay, is applied to the delay of any flight. As such, the model described in Andersson (2001, 2006) does not reflect the standard practices of current commercial airline management.

Andersson et al. (2001, 2004) also developed two other methods were proposed to solve the flight disruption management problem: a Lagrangian heuristic and a Dantzig-Wolfe-based method. The TS method was deemed superior, producing better or almost equally good solutions in shorter times in the majority of the problems studied (Andersson 2006). This bodes quite well for the potential of an implementation of *advanced, state-of-the-art* TS methodologies.

2.4 TABU SEARCH

A complete review of the past literature and applications of TS metaheuristic would be inappropriate, if not impossible, in this dissertation. Therefore, in the following subsections, an introduction and an overview of relevant publications will be presented.

2.4.1 Introduction of Tabu Search

TS is a metaheuristic search method that uses memory structures to direct an efficient and effective search of a solution spaces associated with large complex constrained optimization problems. Extensive and detailed discussions of TS abound. One such discussion is contained in Glover and Laguna (1997). In essence TS starts from an initial solution, defines a "neighborhood" which can be reached from the current solution by a "move," a simple change to the current solution. A move's value is the associated change in the objective function value. One use of the memory structures is to control the search by forbidding tabu moves that would return the search to previously visited solutions for a specific number of iterations, the tabu tenure. Various strategies may be adopted to improve the search. For example, an aspiration criterion can be employed to override tabu restrictions in specified circumstances. Intensification strategies can be used to concentrate the search in the vicinity of "good solutions," while diversification strategies are used to encourage the search to proceed to a different area of the solution space. In adaptive and reactive tabu search (Battiti and Techiolli 1994), search parameters like the tabu tenure are automatically adjusted based on the quality of the search. Adaptive Tabu Search (ATS) myopically decrements (increments) the tenure based on whether the objective function improves (disimproves). Reactive tabu search (RTS) changes the tabu tenure according a more global set of decision rules. In RTS the history of solutions visited is maintained during the search and is used to check if the search has been restricted in an "attractor basin" residing in the solution space. RTS also uses various mechanisms to escape from chaotic attractor basins once they are identified.

2.4.2 Applications of Tabu Search

Laguna et al. (1991) applied TS methods on a single machine scheduling problem. Laguna et al. (1991) described a TS-hybrid method that employs both swap and insert

move. Barnes et al. (1993) solved the multiple-machine weighted flow time problem using static TS. Compared to the branch and bound method, their computational experiments showed that TS is superior to branch and bound in the quality of solutions and the time needed to obtain a solution. Also, there is only a modest growth in the computational effort required to acquire the solution as the number of jobs and machines get larger.

Battiti et al. (1994) presented a reactive TS method, which adapts the size of tabu tenure in response to the search history. The tenure was increased when configurations were repeated and reduced in the absence of such repetitions.

Barnes et al. (1995) applied TS to solve the job shop scheduling problem. Starting from the best solution rendered by a set of 14 heuristic dispatching solutions, it iteratively moves to another feasible solution by reversing the order of two adjacent critical path operations performed by the same machine. Laguna et al. (1995) presented a TS method to solve the multilevel generalized assignment problem which used ejection chains to construct the candidate list of moves at each iteration of our solution approach. Carlton and Barnes (1996) used the reactive TS to solve the TSP with time windows. Their experiments showed that the reactive TS is robust across a wide range of problem types.

Lokketangen et al. (1998) solved general zero-one mixed integer programming problems using TS. González-Velarde et al. (2002) used TS employing ejection chains to solve graph coloring problem. Nanry et al. (2000) used reactive TS to solve a pickup and delivery problem with the constraints of vehicle capacity and customer time windows. Three neighborhood moves were used, with a hierarchical search methodology to dynamically alternate between neighborhoods.

Korycinski et al. (2003) combined TS within a classification algorithm. Reactive tabu search was used to select features in hyperspectral data analysis to improve

classification accuracy. Scrich et al. (2004) applied TS to the problem of scheduling jobs in a flexible job shop with the objective of minimizing total tardiness. Improved solutions were found in neighborhood generated by the critical paths of the jobs in a disjunctive graph representation.

Barnes et al. (2004) used group theoretic TS method to solve the aerial fleet refueling problem. They applied group theory to partitioning and ordering (P|O) combinatorial problems. Combined with dynamic search methodologies, the algorithm was shown to be effective and efficient. Crino et al. (2004) also used group theoretic TS to solve the theater distribution vehicle routing and scheduling problem. Harwig et al. (2006) used an adaptive TS to solve 2-dimensional orthogonal packing problems. Using a very efficient dynamic move neighborhood strategy the method quickly finds excellent near-optimal solutions, Kinney et al. (2007) developed a group theoretic TS algorithm to solve the unicost set covering problem by partitioning the solution space into orbits and a reactive TS procedure based on both inter-orbit and intra-orbit swap was used to explore the neighborhood. Their method outperforms CPLEX on a widely used set of benchmark problems.

Though TS has been applied extensively in various practical problems and successfully attacked those problems, using it in airline disruption management problem has not been extensively and properly studied and modeled. In the next two chapters, efficient methodologies using TS to attack two different airline disruption management problems, the Aircraft Grounding Problem and the Reduced Station Capacity Problem, are presented.

Chapter 3

The Aircraft Grounding Problem

3.1 PROBLEM DESCRIPTION

The aircraft grounding problem (AGP) addressed in this chapter is associated with disruptions caused by aircraft groundings where aircraft are out of service due to unexpected events, such as mechanical failure, or other unforeseen scenarios. Such an unplanned event immediately disrupts the schedule and directly affects those flights assigned to the grounded aircraft. Furthermore, in most cases, the original schedule's required "station balance" will be not satisfied even after the grounded aircraft are once more available, i.e., each station must have the same number of aircraft as would have been present if no disruption had occurred.

Thus, we must revise the current schedule, i.e., alter the aircraft routes (the flight sequences assigned to each of the aircraft). This revision is accomplished by selecting flights to be cancelled or delayed so that the original published schedule is again valid at the end of the recovery window. We desire to minimize the schedule revision cost, including the tangible costs (loss of profit) associated with flight cancellations, and the intangible costs associated with flight delays and with deviations from the original routes. It is important to minimize the changes to the routes because such alterations cause undesired changes to crew and other resource assignments.

Previous research attempted to preserve routes by minimizing the number of flights assigned to different aircraft (Thengvall et al. 2000). As we discussed earlier in Section 2.3, this measure, however, is not consistent with airline practice where deviations from

original routes are measured by the *swaps* of flights between routes. The following simple example illustrates this fact.

Let the original routes of aircraft *A* and *B* be composed of the following flights:

A: 1 2 3 4 5 6

B: 7 8 9 10 11

and after a single swap, the new routes are

A': 1 2 3 9 10 11

B': 7 8 4 5 6.

(The changes are underlined.) However, *6* flights changed aircraft. The following routes are obtained by two swaps, but only 2 flights changed aircraft:

A": 1 2 3 9 5 6

B": 7 8 4 10 11.

Because of fewer manipulations of crew and aircraft required, A' and B' cause fewer disruptions to the routes and are preferred despite having more flights change aircraft.

3.1.1 The Objective Function

The objective function, Z, to be minimized, includes the costs associated with cancellation, delays and swaps:

$$Z = \sum_{j \in J \setminus S} g(\overline{D_j} - D_j) + \sum_{j \in S} C_j + f(n)$$

Where J = set of flight indices

S = set of cancelled flight indices in the revised schedule

C_j = the cancellation cost for flight j

D_j = departure time of flight j in the original schedule

$\overline{D}_j$ = departure time of flight j in the revised schedule

n = number of swaps in the revised schedule

$g(m)$ = delay cost function (minutes, m)

$f(n)$ = swap cost function.

A revised schedule flight is *late* if it departs later than its original scheduled departure time. Jarrah et al. (1993) proposed $20 per minute for *any* delay. This was used by other researchers including Argüello et al. (1997). However, in this research, a *minor* delay cost is incurred for delays less than 15 minutes because such delays are not formally considered to be a *chargeable* delay, according to the Bureau of Transportation Statistics (http://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp). Longer delays are penalized $20 per minute for *all* delay time (including the first 15 minutes). Hence,

$$g(m) = \begin{cases} 20, & \text{if } m < 15 \\ 20m, & \text{if } m \geq 15. \end{cases}$$

Flight cancellation costs, a combination of lost revenue, loss of passenger goodwill and other associated negative effects, are specific for each flight and are predefined for each flight.

Consulting professionals from the airline scheduling industry provided the following information about swaps: The more swaps, the faster the negative impact increases. Therefore, the swap cost function is a convex function suggested by professionals from the airline scheduling industry. The cost of 15 swaps is considered approximately equivalent to:

(1) the average cost of a cancelled flight or

(2) about 6 "short" delays of more than 15 minutes.

Based on the above observations, the swap cost is well approximated by the following polynomial function of the number of swaps, n:

$$f(n) = 11.11\ n^2.$$

The use of a nonlinear function to account for the effect of swaps across routes is entertained for the *first* time in this dissertation.

3.1.2 Constraints

There are 3 principal constraints that must be satisfied: (1) A successful recovery to the original schedule requires that the station balance be restored (Argüello et al. 1997), (2) for the route of each aircraft, the revised schedule also must achieve a minimum turnaround time between the arrival of flight and departure of the subsequent flight (Argüello et al. 1997), and (3) ferry flights (an aircraft flight without passengers) are not allowed.

3.2 A DISCUSSION OF THE METHODOLOGIES AND ALGORITHMS DEVELOPED FOR THE AGP

Three approaches were developed to attack the AGP:

(1) a pure TS approach (AGP-TS),

(2) a hybrid method (HM1) combining a time-space network model with a Limited TS (LTS), and

(3) HM2, HM1 followed by AGP-TS.

3.2.1 A Pure TS to the AGP – AGP-TS

3.2.1.1 The Solution Representation

The AGP solution representation is composed of a list of the revised routes of the aircraft associated with the problem. In this context, a route is an ordered list of flight indices assigned to an aircraft index. A schedule or solution is represented as list of routes. Flight and route indices begin at 0. Flights that are cancelled are assigned to "dummy" aircraft whose indices are equal or larger than the number of available aircraft.

A flight index "flag" of -1 ends a route. For example, consider the following small three aircraft solution where the aircraft index is implicit in the ordering of the routes

1 2 3 -1

5 6 7 -1

9 10 11 -1

0 4 8 -1.

Flights 1, 2 and 3 are assigned to aircraft 0; flights 5, 6, and 7 are assigned to aircraft 1; and flights 9, 10 and 11 are assigned to aircraft 2; flights 0, 4 and 8 are cancelled since they are assigned to dummy aircraft 3.

Given the original schedule with the departure time of each flight provided, the revised routes define the revised flight schedules by setting the new departure time of a flight to be the later of (1) the arrival time of last flight plus turnaround time or (2) the original departure time.

3.2.1.2 The Neighborhood Definitions

There are 4 AGP-TS neighborhoods. Among them (1) and (2) were inspired by Argüello et al. (1997).

(1) Circuit insert*:* insert a *circuit* starting and ending at the same station into an aircraft or dummy route. For example, consider routes A: 0 1 2 3 4 and B: 5 6 7 8 9. If flights 1 and 2 form a circuit starting and ending at flight 6's terminal station, then circuit (1 2) inserted into route B after flight 6 yields A': 0 3 4; B': 5 6 1 2 7 8 9. If, in A, (2 3 4) forms a circuit starting at the terminal station of flight 9, then appending (2 3 4) to the end of B yields A": 0 1 and B'': 5 6 7 8 9 2 3 4.

A circuit may also be inserted into a different position in the same route. For example, given A: 0 1 2 3 4 5 6 with circuit (1 2 3) starting at flight 5's terminus then (1 2 3) can be inserted after 5, yielding A''': 0 4 5 1 2 3 6.

(2) 2 sub-route swap: swap two sub-routes with the same starting stations at the end of two "real" routes, or swap two sub-routes with the same starting stations and ending stations.

For example, consider routes A: 0 1 2 3 4 5 and B: 6 7 8 9 10 11. If flights 1 and 7 start at the same station and flights 3 and 8 terminate at the same station, then (1 2 3) and (7 8) can be swapped yielding A': 0 7 8 4 5 and B': 6 1 2 3 9 10 11.

Consider routes C: 0 1 2 3 4 5 and D: 6 7 8 9 10 11. If flights 3 and 10 start at the same station then (3 4 5) and (10 11) can be swapped, yielding C': 0 1 2 10 11 and D': 6 7 8 9 3 4 5. After such a swap, the ending stations of the two routes are *also* swapped.

(3) Within route swap: swap two sub-routes with the same starting and ending station inside the same route.

Consider route A: 0 1 2 3 4 5 6 7. If flight 1 and 4 start at the same station, and flight 2 and 6 end at the same stations, then (1 2) and (4 5 6) can be swapped yielding A: 0 4 5 6 3 1 2 7.

(4) Cancel 2 sub-routes: cancel two ending sub-routes.

Consider routes A: 0 1 2 10 11 and B: 6 7 8 9 3 4 5. If flights 2 and 5 end at the same station, and flights 9 and 11 end at the same station, then (10 11) and (3 4 5) may be canceled yielding routes A: 0 1 2 and B: 6 7 8 9 and creating dummy routes C: 10 11 and D: 3 4 5.

3.2.1.3 Move Evaluations

The *move value*, the difference between the current, *incumbent*, solution's objective function value and that of a neighboring solution must be computed for all neighboring solutions. For efficiency only changes in the routes affected by the move are

included in that computation, significantly reducing the effort required for move evaluation.

3.2.1.4 The AGP-TS Attributes

The following two types of TS attributes were used in the experimental studies associated with the development of AGP-TS. The second was found to provide superior performance and was used in all subsequent work.

A Sub-route First Flight Tabu Attribute

If a flight is moved, it is forbidden to be moved again, *as the first flight* in a candidate sub-route, for tabu tenure iterations. A vector is employed where the value of *tabu_list[i]* indicates the earliest iteration at which the flight *i* may again be moved to *any* other position, as the first flight in a candidate sub-route. Each time a candidate move is considered, *only* the tabu status of the flight at the *beginning* of the sub-route to be moved is checked. Each time a move is executed, the *tabu_list*[] for all flights moved is updated.

A Route-Position-Flight Tabu Attribute

Suppose flight *i* is at position *j* of route *k* and it is moved as the first flight in a sub-route relocation. That sub-route may not be returned to *position j* of route *k* for tabu tenure iterations.

For example, suppose, in the current solution, route A is (0 1 2 3 4) and route B is (8 9 10 11). An insertion of sub-route (1 2 3) to route A at position 2 yields routes A': (0 4) and B': (8 9 1 2 3 10 11). Sub-route (1 2 3) may not return to position 1 in route A for tabu tenure iterations. The data structure employed not only prevents direct moves from causing this result but also prohibits other moves from indirectly causing this condition.

3.2.1.5 The Tabu Tenure

In the AGP, the problem size is directly correlated with the number of aircraft. Since large AGPs usually need large initial tabu tenures, the initial tabu tenure is defined as the total number of aircraft multiplied by a coefficient. In the research documented here, a coefficient of 0.7 was used. This was determined through empirical studies and was held constant for all problems. Based on empirical tests, the upper bound of the tenure is set to the maximum of (*initial tenure* · 1.7) and (*initial tenure* + 5), the lower bound is given by

$$\text{MAX}(\ \text{MIN}(\ \textit{initial tenure} \cdot\ 0.5,\ \textit{initial tenure} - 5\ ),\ 2\ ),$$

which assures that the tenure never is less than 2. Thus, the upper bound and the lower bound are proportional to the problem size and stay within a reasonable range.

Reactive TS was employed and the tabu memory structure was extensively used to control the search and to adjust the search parameters based on the quality of the search. The search quality is determined by the frequency of revisiting previously visited solutions. In the AGP, the solution is represented by the routes of aircraft. The simplest way to identify the solution is to compare the routes with routes of all previously visited solutions. However it is time-consuming and memory-consuming considering the number of solutions visited and the size of the solution. In order to identify previously visited solutions efficiently, a two-level comparison mechanism was used.

The solution history is composed of solution-identity information and visit information. The solution identity information includes the objective function value and its hash value. The hash value is calculated from its route as follows:

$$Hash(solution) = \sum_i \sum_j solution.routes[i][j]\ \cdot\ prime\_num[i].$$

The visit information includes the number of repeated visits and the iteration that each of the previous visits occurred.

Because different solutions may have the same objective function value, the objective function value alone cannot be used to uniquely identify the solution. The hash value, computed from the route, uniquely identifies the solution when the objective function values are the same. The procedure is as follows:

(1) After finding the incumbent solution *s*, calculate its hash value, *Hash(s)*.

(2) Search for the objective function value of the solution *s* in the solution history. If it is not found, then this solution has never been visited and the solution *s* is added to the solution history. Otherwise, among all solution history records with this objective function value, determine if the hash value *Hash(s)* is already present. If not, *s* has never been visited and its hash value is added to the solution history. If found, *s* is being revisited. Update the revisit information.

As recommended by Battiti and Techiolli (1994), tabu tenure is adjusted in the following way: If a solution is revisited within a specified number of iterations (CYCLE_MAX), then tenure is increased by a predetermined factor to diversify the search. A moving average of the iteration intervals between the solution revisits is calculated to track the recent revisitation cycle length in the search history. If tabu tenure has not been increased for more iterations than this moving average, then tabu tenure is decreased to avoid excessive increase in tenure and to intensify the search. Finally, when AGP-TS determines that all possible moves are tabu and none satisfy the aspiration criterion, then the tabu tenure is decreased, with the first solution on the elite list of solutions selected as the new incumbent solution and the tabu memory structure is reinitialized.

As discussed in Section 3.2.1.6, a mechanism is also implemented to escape from an attractor basin.

3.2.1.6 The AGP-TS Algorithm

Constructing the Initial Starting Solutions

Routes in an initial starting solution may extend beyond the end of the operation day. Such infeasibilities were always corrected by the subsequent TS methodology.

In constructing an initial solution, we rearrange the flights previously assigned to the grounded aircraft routes while satisfying station aircraft balance. Those flights are either cancelled or appended to the routes corresponding to the ungrounded routes.

In overview, we first cancel grounded routes which start and end at the same station and cancel route pairs that "exchange" the starting and ending station, *i.e.*: route i (j) starts (ends) at station A and ends (starts) at station B.

For the remaining routes assigned to grounded aircraft, we combine them as much as possible, and then append them to the end of *applicable* routes of ungrounded aircraft. The reason for combining the routes first is that the "applicable routes of ungrounded aircraft" may be used up if we try to append each route of a grounded aircraft individually to the route of ungrounded aircraft.

Often, there are multiple choices for the sequence of routes to combine and the selection of routes to append to ungrounded aircraft routes. Thus different initial solutions can be generated. The pseudocode for constructing the initial starting solution is presented in Appendix A.1.

An Overview of AGP-TS

AGP-TS begins from an initial starting solution. The search and its result will vary with each initial starting solution. However, experiments showed no correlation appears to be present between the quality of the initial starting solution and the quality of

the best solution obtained by TS. For this reason the starting solution with the median objective function value is used.

AGP-TS maintains a memory structure which records attributes of the solutions encountered, including the objective function value, the unique hash value of each solution to identify individual solutions, and the iteration number(s) in which the solution was visited. It also maintains an elite solution list of good solutions. The memory structure is used to determine if search is trapped in an attractor basin.

The corresponding parameters in RTS are defined as follows:

REP = 3 (number of repetitions to be considered as "frequent" solution)

CYCLE_MAX = 10 (If a solution is repeated in less than 10 iterations since the last repeated solution, increase the tabu tenure)

CHAOS = 2 (number of frequent solutions to trigger an escape).

The history of solutions visited is maintained during the search. If 2 solutions are visited more than 3 times each in the recent search history, the search is said to be trapped in an attractor basin. In this case, an escape process is performed clearing all tabu memory structures and a sequence of escape moves are performed to lead to a markedly different region of the solution space. In this problem, three different escape mechanisms were implemented and tried:

(1) Perform the most disimproving neighborhood move for a specified number of iterations;

(2) Perform the first non-tabu move in the neighborhood of the incumbent solution for a specified number of iterations;

(3) Perform moves that un-cancel currently canceled flights for a specified number of iterations.

The experiments performed with these three mechanisms clearly indicated that the first strategy was superior and it was used in all later computational experiments.

The pseudocode of the main AGP-TS program is presented in Appendix A.2.

3.2.2 A Hybrid Methodology of Time-Space Network Flow Model and TS (HM1)

First, in HM1, a time-space network is constructed. After solving the network flow problem with CPLEX, the obtained arc-based solution is post-processed to generate a route-based solution. Next, a LTS procedure is used to find improved route-based solutions with less swaps between aircraft since the single commodity network flow model does not generate routing information and therefore cannot model the "swaps."

Figure 3.1 shows the steps performed by HM1. First a time-space network is generated with the given delay options. Then the LP relaxation problem is solved by CPLEX. Since the solution is arc-based without the route information, a post-processing procedure is implemented to generate a route-based solution using a one-pass scan of the network and removing the extra delays caused by the discretization. Keeping the flight departure/arrival times unchanged (thus freezing the delay and cancellation costs), the number of swaps (and their cost) is then decreased by LTS to find a better route-based solution. The total objective function value is the sum of the cancellation, delay and swap costs.

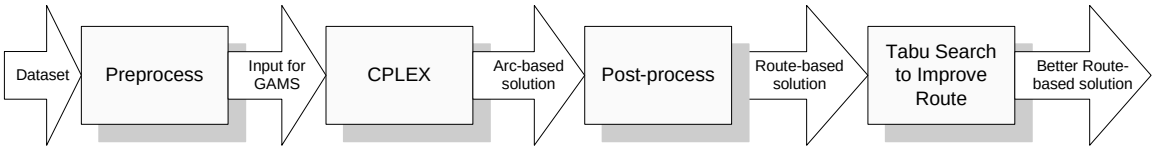

Figure 3.1: Solve by Network Flow Model: Flow Diagram

3.2.2.1 Time-Space Network Model

The time-space network flow model, a network flow problem with side constraints, was initially presented by Yan and Yang (1996) and is known to be NP-hard. Thengvall et al. (2000) derived their model from the work of Yan and Yang (1996) to address the AGP.

In this model, as illustrated in Figure 3.2 (excerpted from Thengvall et al. 2000), each node is associated with a station and a specific point in time, where time flows downward in the network. The time horizon extends from the start to the end of the operation day, which defines the end of the recovery window. Aircraft flow on the arcs. The diagonal arcs allow fights between two stations and the vertical arcs represent aircraft waiting at a station for the next flight.

Delays are incorporated by adding nodes and arcs associated with the different specified delay periods. Figure 3.3 (Thengvall et al. 2000) used two delay options (i.e., 10 minutes and 30 minutes) for each of the four fight legs. With more delay options, more delay arcs are added to the model and thus more nodes are added.

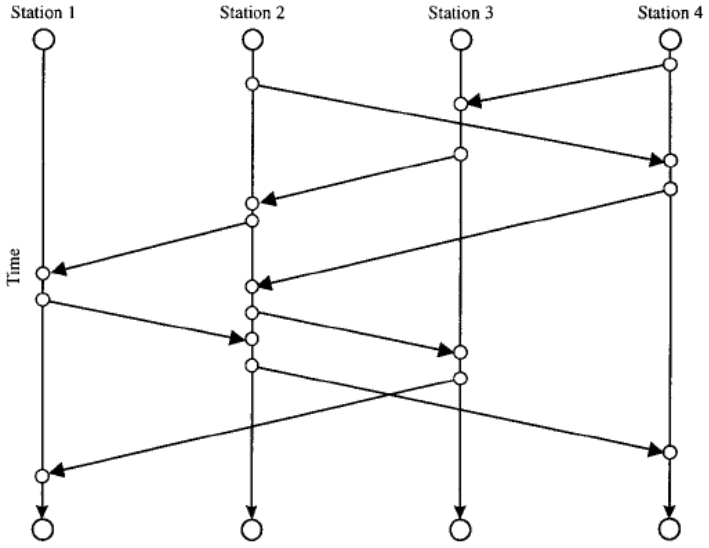

Figure 3.2: Time Space Network Representation

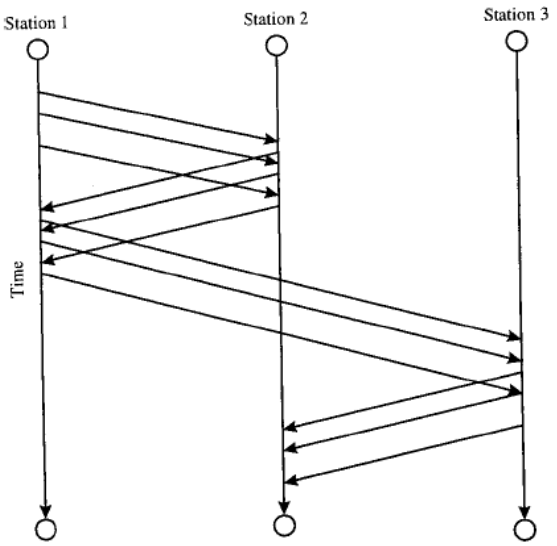

Figure 3.3: Incorporate Delays

Each flight is covered by either the original flight arc or a delay arc or the flight is cancelled. This corresponds to a set of binary variables for which exactly one is assigned a value of 1.

The network is generated from the original schedule with a specified set of delay options, i.e., all 5 minute intervals from 5 to 360 minutes, comprising 72 possible delay options. First all nodes and arcs associated with the original flight schedule are added. Then the nodes and arcs associated with each delay option for each flight are added. Finally for all stations, the ground arcs are added between each pair of adjacent nodes associated with the same station. The mathematical time-space network flow model is

Indices and sets:

n: node

a: arcs

f: flights

S: supply nodes

D: demand nodes

$I(n)$: arc set entering node n

$O(n)$: arcs set leaving node n

$Z(f)$: arc set for flight f

Parameters:

C_a: cost of arc a

s_n: supply at node n

d_n: demand at node n

β_f: cancellation cost of flight f

Variables:

x_a: flow on arc a (integer)

y_f: =1 if flight f is cancelled

Minimize $\sum_a C_a x_a + \sum_f \beta_f y_f$

Subject to:

$$\sum_{a \in I(n)} x_a = d_n, \forall n \in D \quad (1)$$

$$\sum_{a \in O(n)} x_a = s_n, \forall n \in S \quad (2)$$

$$\sum_{a \in I(n)} x_a = \sum_{a \in O(n)} x_a, \forall n \in N \setminus (D \bigcup S) \quad (3)$$

$$\sum_{a \in Z(f)} x_a + y_f = 1, \forall f \in F \quad (4)$$

$$y_f \in \{0,1\}, \forall f \in F \quad (5)$$

$$x_a \in \{0,1,...\}, \forall a \in A. \quad (6)$$

The objective is to minimize the sum of cancellation and delay costs. Constraints (1) and (2) enforce the station balance of aircraft. The flow balance at intermediate nodes is enforced by constraint (3). Flight coverage is ensured by constraint (4).

The model was implemented using GAMS and solved by the CPLEX 9 solver. As a first step, constraints (5) and (6) were replaced with

$$y_f \geq 0, \forall f \in F \tag{5a}$$

$$x_a \geq 0, \forall a \in A \tag{6a}$$

to create an LP relaxation of the original integer problem. If the relaxation, when solved by the CPLEX LP solver, yields an integer solution, we have found the optimal solution to the original integer problem. Otherwise, the CPLEX MIP solver is called to solve the original integer problem. After an integer solution is obtained, a simple one-pass algorithm, presented in the next section, constructs the associated routes.

3.2.2.2 A One-Pass Algorithm to Construct the Routes

A one-pass algorithm is used to create a route solution from the arc-based solution which becomes the starting solution used by the later TS algorithm to improve the solution. The one-pass pseudocode is given in Figure 3.4 where the arcs are searched in their indexed sequence.

```
Input: the network, the flows on the arcs
Output: the routes

For (each supply node n_s with nonzero supply)
{  Let n = n_s;
   while ( n is not a demand node )
   { Search all arcs to find arc a, where (flow on a > 0 ) ∩ ( a starts from n )
     Decrease the flow on a by 1
     n = end node of a
   }
    A route has been finished; decrease supply at n_s by 1
}
```

Figure 3.4: One Pass Procedure to Generate Route

Therefore, in the route generation algorithm, when choosing an arc from the arcs originating from node *n*, flight arcs have higher priority and the flight arc with the earliest departure time is picked first.

3.2.2.3 Finding a Better Route-based Solution Using LTS

Since, given an arc-based solution, there could be multiple corresponding route solutions, a LTS procedure was constructed to lessen the number of swaps in the route solution. The route solution generated from the one-pass algorithm provides the initial solution for the LTS procedure.

The number of swaps associated with a new solution, when compared to the original routes, is determined in the following way:

Define *o(f)* to be the route which flight *f* belonged to in the original routes. Now, for all new solution routes, i.e., for each route *r*, go through each flight *f* assigned to *r* and count how many times *o(f)* changes value. The total number of changes is the swap number for the new route solution.

The LTS Neighborhood Definitions and Memory Structure

The solutions in the neighborhood of the incumbent solution do not alter the flight cancellations associated with the current schedule, neither the departure and arrival times of the flights. However, the sequence of flights in the routes may be altered. In the following descriptions, a departure time is consistent with the current initial solution.

(1) A swap neighborhood

Swapping two sub-routes at the end of two routes is allowed if it does not increase the departure time of the first flight in either sub-route.

For example, two routes are given as below.

A: 1 2 3 6 18

B: 9 11 12 7

Suppose that flights 18 and 7 depart from the same station. If the current departure time of flight 7 is not earlier than the arrival time of flight 6 plus the minimum turnaround time, and the current departure time of flight 18 is not earlier than the arrival time of flight 12 plus the minimum turnaround time, then flights 18 and 7 may be swapped yielding

A: 1 2 3 6 7

B: 9 11 12 18.

(2) An append neighborhood

A sub-route may be removed from its original route and appended to the end of another route if it does not delay the current departure time of the first flight.

For example, two routes are given as below.

A: 1 2 3 6 18

B: 9 11 17.

If flight 18 departs from the same station as flight 17 arrives, and the departure time of flight 18 is no earlier than the arrival time of fight 17 plus the minimum turnaround time, then flight 18 may be removed from route A and appended to the end of route B, leading to the following result:

A: 1 2 3 6

B: 9 11 17 18.

In the above two neighborhoods, the station balance is maintained and the move value is simply the change in swap number.

The tabu attributes are defined as route-position-flight tabu attribute described in Section 3.2.1.4.

The TS approach described in this section is quite similar to a greedy search using the same neighborhood definitions. Due to the limited solution space, in all the problem instances that we tested, TS found the same solution as greedy search.

3.2.3 Another Hybrid Methodology of Time-Space Network Model and TS (HM2)

In this approach, the result from HM1 is used as initial starting solution for the AGP-TS described in Section 3.2.1. The tabu attributes, neighborhood definition and search mechanism are the same as the TS of Section 3.2.1. The idea is to investigate whether the HM1 result can be improved. The computational results are presented in Section 3.3.

3.2.4 A Multicommodity Network Flow Model

3.2.4.1 The Basic Multicommodity Network Flow Model

In order to include the aircraft assignments in the model, a multicommodity network flow can be formulated for our flight rescheduling problem. In this model, each aircraft is treated as a separate commodity which flows from the supply node to one of the demand nodes. Side constraints ensure the coverage of each flight. The values of the decision variables, the flows on each arc for each aircraft, indicate the assignment of the flight associated with each arc.

Similar to the previous described time-space network model, each node is associated with a time and station. There are two types of arcs. All of the diagonal arcs represent flights, either on time or delayed. Given n delay options, each flight is associated with $n+1$ parallel flight arcs. The vertical arcs represent the aircraft waiting at the station. The multicommodity network differs from the single-commodity time-space network is that now each of the arcs are *duplicated a* times where a is the number of aircraft.

Figure 3.5 presents a multicommodity network with three stations and two aircraft. Only two delay options (0 minutes and 30 minutes) are allowed. Between each pair of connected nodes, there are two arcs, each associated with one of the aircraft. The nodes n_1 and n_3 are supply nodes for aircraft *1* and aircraft *2*, respectively. The nodes n_2 and n_6 are the demand nodes. The red lines indicate aircraft *1* arcs and the green lines indicate aircraft *2* arcs. Bold arcs have the flow of 1 and the network flows stipulate a unique route assignment for each aircraft.

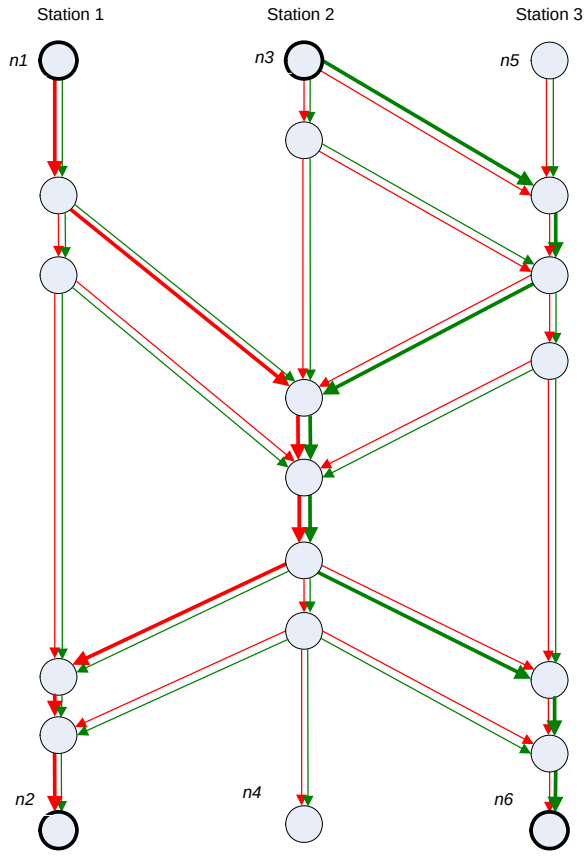

Figure 3.5: Multicommodity Network

The number of swaps relative to the original solution routes is identical to the earlier definition given for the single commodity network. In this model, we define swap

penalty for n swaps to be $penalty(n) = k \cdot n^p$, where $k>0$ and $p>1$ are predefined coefficients.

In the multicommodity network flow model, the number of swaps number is more difficult to model. In the following section, a lower bound on the number of swaps is presented. Next we present a formulation of the number of swaps by combining the time and station information into each node in the network.

3.2.4.2 A Lower Bound on the Number of Swaps

A lower bound on the number of swaps may be determined by considering how many flights are assigned to different aircraft when compared to the original schedule. If, in the original route, flight f_1 is followed by f_2 and now the two flights are assigned to different aircraft, then apparently a swap occurred. The converse is not true because scenarios may exist where the two flights are assigned to the same aircraft but are not adjacent. Therefore, the swaps counted in this way provide only a lower bound on the actual number of swaps. A mathematical model based on this lower bound could be presented but it would have little practical value.

3.2.4.3 A Model to Precisely Count the Number of Swaps

To model number of swaps, we need to record the associated time and station for each node in the network, i.e., each node has a *station index* and *timestamp* associated with it. In addition, each ground arc is associated with one station and each flight arc is associated with a specific origin and terminal station pair. A complete mathematical formulation of the multicommodity network associated with this model is given in Appendix B.

Unfortunately, the problem size grows exponentially with the increase of aircraft, flights and number of delay options and the problem. This problem is very difficult to solve due to its complicated problem structure with a non-linear term.

Experiments were conducted trying to solve the multicommodity network model where the objective function included a simplified quadratic term that mimicked the swap costs. For a problem for the 737 dataset with 27 aircraft and 162 flights, and a time discretization of only 9 delay options, CPLEX reported that the IP with 23,298 rows, 47,274 columns and 119,607 non-zeros. CPLEX failed to solve this QP model.

3.3 COMPUTATIONAL EXPERIMENTS

This section presents the comparative computational results obtained from experiments with three methodologies: AGP-TS, HM1 and HM2.

3.3.1 The Experimental Dataset

Experiments were performed using the 737 dataset from Argüello (1997). It is composed of 162 flights covering one operation day by 27 aircraft among 30 airports. Thirty minutes of minimum turnaround time is required between the arrival time and the subsequent departure in the aircraft route. It is a hub-spoke schedule with station EWR as the hub.

The disruption is grounding one or more aircraft at the beginning of the operation day, and the recovery window embraces the entire operation day.

3.3.2 The Comparative Computational Results

The three algorithms were coded in C++ and run on Dell Precision 530 Workstations running SuSE Linux with two 1.8GHz Pentium Xeon processors utilizing 1GB of RAM.

The problem instances solved were *randomly* selected from the larger set considering the grounding one, two, three and four aircraft among 27 aircraft. For each number of grounded aircraft, 15 problem instances are randomly selected yielding a total of 60 problem instances. This subset of all possible combinations are selected so that a proper size of problem pool can be used to compare results from those three methodologies.

In applying the hybrid methods to all 60 problem instances, the CPLEX LP solver yielded integer solutions while ignoring swaps between routes implying that each problem possessed a totally unimodular constraint structure.

In applying the TS approach, 200 iterations were allowed in each run. Experiments showed that for most of problem instances, TS was able to find a solution with comparable objective function value within 200 iterations.

Table 3.1 shows the average results obtained from the three algorithms.

	AGP-TS	**HM1**	**HM2**
Average obj.	33981	32002	31268
Average cancellations	13.9	11.1	11.1
Average cancellation cost	32048	23268	23447
Average delay cost	1463	5863	6057
Average swaps	5.7	15.1	12.1
Average Time (seconds)	46	298	298+15
Time find the best (seconds)	27	-	-

Table 3.1: Comparison of Computational Results from Three Algorithms for AGP

3.3.2.1 AGP-TS vs. HM1

In the 60 problems solved, HM1 (AGP-TS) obtained a better objective function value than AGP-TS (HM) in 56 (4) problems. From Table 3.1, HM1 has both lower cancellation costs (tangible costs) and a lesser number of cancellations, while AGP-TS has lower delay and swap costs (intangible costs). It should be noted that the simultaneous consideration of the solutions from the AGP-TS approach and from HM1 and HM2 can be quite beneficial since any of the three solutions could be preferable depending on a decision maker's preferences and priorities. The individual results for each problem are given in Table C.1 and Table C.2.

Table 3.1 shows the average computation time used for each method. The times used have small differences from problem to problem. In these experiments, AGP-TS had an average of 46 seconds for 200 iterations, and the time AGP-TS spent to find its best solution was, on average, 27 seconds. HM1 used an average of 298 seconds and about 296 seconds were required for CPLEX to solve the LP relaxation of the problem.

HM1 obtains better overall objective function values than AGP-TS. In stage 1, HM1 achieves *the* global minimal solution when only the sum of cancellation and delay costs is considered (in its first stage). After using LTS to minimize the swap costs, the total objective is usually still better than AGP-TS which tries to solve the problem in a holistic fashion.

3.3.2.2 HM1 vs. HM2

As we stated earlier, the HM2 method simply applies the AGP-TS using the result from HM1 as the starting solution. The idea behind HM2 is to determine if AGP-TS can improve the result from HM1.

Out of the 60 problems, 48 showed improvements. Among them 13 showed changes in cancellation cost though the total number of cancellations remained unchanged.

In these experiments, tabu search either in the limited form for HM1 or the "full" form for HM2 contributed markedly to the solution of the problems in terms of reducing the final objective function value.

3.3.2.3 The Termination Conditions for the AGP-TS Module

Experiments with the AGP-TS algorithm have been performed to investigate the effect of allowing an increased number of allowed iterations. Three termination conditions were studied: (1) 200 iterations, (2) 2000 iterations and (3) terminating the search if 100 iterations are performed without improvement in the objective solution found or if 2000 iterations are reached.

Table 3.2 presents the results for the 60 problem instances using these termination conditions. Twenty-five problem results improved when 2000 iterations were allowed (41.7% of the 60 problem instances). Three problem results were improved by 10% or more. The dynamic termination condition did not improve the results but did require less time than the 2000 iteration termination condition.

	200 iterations	2000 iterations	Dynamic Termination
Average obj.	33981	33400	34403
Average cancellation numbers	13.9	13.7	13.8
Average swap numbers	5.7	5.7	4.6
Average Time	46	622	48
Time find the best (seconds)	27	86	22
Number of problems Improved (compared to 200 iterations)	-	25 (41.7%)	10 (16.7%)
Number of problems Improved 10% or more (compared to 200 iterations)	-	3 (5%)	0 (0%)
Max/Min Number of Iterations Run	200	2000	354/101

Table 3.2: Comparison of Different Termination Conditions for AGP-TS

3.3.2.4 Experimental Comparisons of TS and GRASP

Experiments were performed to compare TS and GRASP (Argüello et al. 1997) for the flight rescheduling problem without considering keeping the original route.

Argüello et al. (1997) solved the flight reschedule problem using GRASP without considering the deviation from the original route. The cost to minimize is the sum of delay cost and the cancellation cost. Since GRASP did not measure the deviation of the revised schedule from the original schedule, it did not consider route deviations.

In order to perform a comparison between GRASP and TS on this problem, we derived the GRASP algorithm based on the descriptions in Argüello et al. (1997) and used the TS algorithm, derived from AGP-TS, to attack the same problems described in

Argüello et al. (1997). The algorithms were coded in C++ and run on a Dell Precision 530 Workstation running SuSE Linux. For both algorithms, the same initial feasible solutions were used to start the search. For each initial feasible solution, 10 seconds of search were allowed.

In the GRASP method, the configuration of updating the solution was set as follows: Only those solutions better than the incumbent can be added to solution list. The solution list is restricted to no more than 10 items. If the list is already full, then every time when a new solution is trying to get into the list the least favorable solution is popped out.

3.3.2.4.1 Comparative Computational Results with the 757 Flight Schedule

The 757 flight schedule (Argüello et al. 1997) comprises a small dataset containing 42 flights, 16 aircrafts and 13 airports. Following Argüello et al. (1997), the following parameters are used: A minimum turnaround time of 40 minutes is required, the cost for a late departure is $20 per minute and the cancellation cost is uniquely given for every flight in the schedule. The disruption to be resolved is grounding one through five aircraft at the beginning of the operation day, and the recovery window embraces the entire operation day.

For this dataset, all 6884 problem instances of grounding one, two, three, four and five aircraft are solved. Table 3.3 presents the comparison of the average results from TS and GRASP algorithms. TS obtained superior results for 2870 problems (41.7%); GRASP obtained superior results for 3792 problem instances (55.1%) and 222 problem resulted in a tie (3.2%). Table 3.3 summarizes these comparative results.

	TS	GRASP
Average Objective Value	52159	51747
# of Problem Obtained the Better Obj.	2870 (41.7%)	3791 (55.1%)

Table 3.3: Comparison of Result from TS and GRASP for 757 Dataset

3.3.2.4.2 Comparative Computational Results with the 737 Flight Schedule

This dataset (Argüello 1997) is larger than the 757 dataset and is composed of 162 flights, 27 aircraft and 30 airports. All 378 problem instances of systematically grounding one or two aircraft are solved.

Table 3.4 presents the comparative average results for the TS and GRASP approaches. TS obtained superior solutions for 290 problems (76.7%); GRASP obtained superior solutions for 88 problems (23.3%). The average objective function value from the TS method was 29214, which was lower than that of GRASP at 31249.

	TS	GRASP
Average Objective Value	29214	31249
# of Problem Obtained the better obj.	290 (76.7%)	88 (23.3%)

Table 3.4: Comparison of TS and GRASP for 737 Dataset

3.3.2.4.3 Concluding Remarks

For the small dataset 757 data, GRASP outperformed TS. However, for the larger 737 dataset, TS outperformed GRASP. This is typical of many cases when TS is compared to GRASP, since TS has access to superior strategies to improve the search. In smaller problems GRASP may be more handy due to the limited search space. However,

with the problem size increases, TS is more powerful in search by avoiding revisiting solutions and therefore being able to investigate more solution space.

3.4 CONCLUSIONS

The goal of the research efforts reported in this chapter was to investigate the use of TS in AGP. Three approaches incorporating TS were implemented and compared: (a) using a AGP-TS, a pure TS, to solve the problem, (b) using HM1, a two step approach consisting of a time-space model to solve the network flow problem, while considering only cancellation and delay costs, followed by a limited TS method that attempts to reduce the route swap costs while using the routes from the first step as a starting solution, and (c) using HM2, which consists of applying the AGP-TS method while starting with the solution obtained by HM1. Experiments on the 737 dataset showed that AGP-TS finds a solution in a relatively short time which has a somewhat greater objective function value than the other methods, while HM1 obtains solutions with superior objective function values than AGP-TS in most of the problem instances tested. 80% of the best solutions from HM1 were improved by HM2. The solutions found by the AGP-TS method possessed markedly different characteristics than the solutions generated by HM1 and HM2 where the AGP-TS method yielded solutions with more flight cancellations and less delay and swap costs.

A multicommodity network flow model that explicitly considered aircraft routes and swap costs was also formulated to address the AGP. The single commodity network flow was not able to address either issue. The CPLEX solver was not able to solve such a multicommodity network flow problem with nonlinear term in objective function.

The results of Chapter 3 naturally lead to the question of why HM1 outperforms AGP-TS in the instances tested. HM1 first only addresses the first two parts of the objective function (cancellation and delay costs) within a network flow model. Only in

step 2 is the third part of the objective function (number of route swaps) separately addressed. Previous research with TS on other problems would indicate, since AGP-TS approach attacks the whole problem simultaneously, that it would be more likely to obtain, overall, solutions with better objective functions. In the remaining part of this section, we discuss the reasons why this perception is incorrect for the AGP.

The possible TS neighborhoods available for the AGP are very limited due to the great difficulty of returning to a feasible solution if the search ever departs from feasibility. The original schedule defines the departure/arrival station and time for each flight. After a disruption occurs, uncancelled flights must still meet the station and time stipulations, such as sufficient turnaround time. Flights may be delayed after a disruption but no flight may take off earlier than its originally scheduled time. Last, and most restrictive and important, is that station balance must be ensured for the next day's schedule. Experiments were performed with neighborhoods that allowed traversal of regions of the solution space where station balance was violated. A return back to station balance was found to be extremely difficult, even if high penalties were dynamically imposed on out-of-balance solutions.

Like the disjoint feasibility region structure described by Van der Bruggen, Lenstra and Schuur (1993) for the vehicle routing problem with time windows, given the polynomial sized neighborhoods that were used, it is likely that the solution space, relative to those neighborhoods, consists of disjoint feasibility subregions. Empirical experiments were performed by starting the AGP-TS method with elite solutions from the HM1 approach. The TS method was never able to move to a region of the solution space resembling the solutions garnered by the AGP-TS approach, even when very long run times were allowed.

Path relinking experiments (Glover et al. 2000) were also performed. The major differences between solutions from the HM1 approach versus the AGP-TS approach are that HM1 solutions have lesser cancellation costs while TS has lesser delay and swap costs. It was conjectured that there might exist hybrid solutions along relinking paths that would capture good properties from both the HM1 and AGP-TS solution types. Construction of relinking paths was attempted in each of the two possible directions.

First, the result from AGP-TS was the starting solution and the result from HM1 was the target solution. Tabu search was conducted to see if the target solution could be reached. Considering that HM1 has a lesser number of cancellations, an auxiliary objective function was imposed which ignored delay costs while calculating move values to encourage decreasing cancellation costs. Extended TS runs were unable to reach the target solutions, implying a lack of connectivity between the starting and target solutions.

The next thing attempted was to use the result from HM1 as the starting solution and the result from AGP-TS as target solution. The time-space model's objective function was modified to minimizing the difference between the starting and target solutions' objective function values. The new problem resulted in a network flow problem with side constraints, with new constraints added to address the difference from the given objective value. The relaxed LP was solved by CPLEX LP solver returning fractional results, indicating that the aforementioned total unimodularity property of the original problem was destroyed by the added constraints. Using the CPLEX MIP solver, instead of the LP solver, yielded a solution with the same objective function value as the target solution. However, the solution obtained was markedly different from the target solution, still possessing lesser cancellation costs with higher delay and swap costs. All of these experiments indicate that it is impractical to find the path from the HM1 solution to the AGP-TS solution by using GAMS/CPLEX.

The HM1 method solves the problem in two stages. In the first stage, the network flow model is solved, yielding the optimal solution for the simplified problem which considers cancellations and delays but ignores swaps. Due to their totally unimodular constraint systems, the LP relaxed problems can be efficiently solved to obtain integer solutions. In the second stage the number of swaps is decreased by LTS. In this stage it decreases the number of swaps by approximately 50% when compared to the route generated by the one-pass route-generating procedure.

In the next chapter, a related problem to the AGP is discussed, the Reduced Station Capacity Problem.

Chapter 4

The Reduced Station Capacity Problem

4.1 PROBLEM DESCRIPTION

One of the most frequent disruptions for airlines is the restriction of maximum number of aircraft on the ground (MOG) during periods of time at one or more stations. This reduced station capacity problem (RSC) can be the result of several situations including reduced gate availability, wing de-icing capacity during a snow storm and runway closures. In these scenarios, the station capacity assumed during the earlier planning phase is not longer available and the airline is forced to reduce the MOG for a particular period of time.

Lesser gate availability may cause reduced MOG. Gates are rented from the airport authority and the rental fee can total millions of dollars per year for a terminal building in a hub airport. When an aircraft incurs unexpected maintenance, fewer gates are available since short-term maintenance is performed at the gates. Inclement weather causes airport congestion and MOG must be decreased to allow the aircraft to be correctly positioned for departure. Inefficient handling of such situations can cause excessive passenger runway wait times and can also result in a *gridlock* situation where incoming aircraft are forced to wait on their assigned runways after landing until sufficient outgoing aircraft leave their gates. This gridlock can quickly generate a "domino effect" in the network causing insurmountable difficulties.

A snowstorm can force the airline to de-ice aircraft on the ground awaiting departure. De-icing requires considerable effort and time which disrupts the flight

schedule because the affected aircraft must wait in line to be de-iced. This can cause delays of multiple flights. Since the accumulation of snow is a function of the time the aircraft spend on the ground, the airline must react by restricting the number of aircraft on the ground such that the total accumulation of snow can be reduced to manageable de-icing levels. Rather than allowing additional arrivals to the reduced MOG station which would exacerbate the problem, it is far better to delay departures of future arriving aircraft at their flight origination stations.

In the research documented in this chapter, we consider the less extreme cases where reduced MOG disruptions are recognizable well in advance, allowing managers to formulate timely flight schedule revisions to manage the forthcoming reduced MOG disruption which involve only a *single* reduced MOG station. Further, we assume that the following parameter values are known: the reduced MOG time period, $[T_s , T_e]$, with start time, T_s , and the end time, T_e, where no more than M aircraft are allowed on the ground. For example, for a future 8 hour period of reduced station MOG, we might assume that we are aware of this 6 hours in advance of MOG reduction (denote this first awareness time to be T_S) and that it is necessary that the original schedule be restored no more than 6 hours after the reduced MOG period ends (denote this required restoration time to be T_E). Therefore, the flights departing within this 20 hour *recovery window*, $[T_S, T_E]$, may be rescheduled. Let us denote such flights as *recovery window flights*.

The above assumptions exclude the most extreme case where all aircraft are forced to immediately leave the station. This type of scenario is *rare* and happens only when the station's very existence is threatened by an imminent threat of severe weather with some level of predictability like an approaching hurricane. After such extreme threatening conditions are no longer present and *if* the station is immediately reopened,

the station balance may be restored in an acceptable time frame through the use of resumed flights and ferry flights.

As described earlier, an airline schedule may be represented as a network where aircraft arrive at and depart from each station at different points in time. At any time, *t,* the number of aircraft on the ground at any station is known. The problem is to reschedule the flights so that no more than *M* aircraft are on the ground at the reduced MOG station at any $t \in [T_s, T_e]$.

The objective is to reschedule the flights to satisfy the reduced MOG constraint in such a way that the disruption costs are minimized while assuring that the original flight schedule including station balance is reestablished within the recovery window.

To address this objective, the disruption cost must be quantified. Just as in the aircraft grounding problem, the cost associated with the disruption can be determined by evaluating the costs of the flight cancellations, flight delays and the swaps of the original routes.

To satisfy the reduced MOG restriction, airlines may choose to delay flights, cancel flights, ferry aircraft, divert flights to other stations and reassign flights to a different aircraft. Delaying flights originally scheduled to arrive at the reduced MOG station before their departures, or canceling flights which were scheduled to fly to the reduced MOG station would cause the number of aircraft at the station to decrease. However, this could introduce station imbalances and thus hinder return to the original schedule after the reduced MOG constraint expires. Swapping aircraft and using ferry flights could aid in restoring station balance at the end of recovery window.

Diverting aircraft to a station different from its original destination will not be considered in this research because the final authority for making a diversion decision resides with the aircraft captain and, thus, can not be externally imposed (Pachon 2007b).

Ferry flights, where aircraft are flown without passengers to reposition aircraft is a quick way to correct station balance but the associated high cost prohibits such flights in all but the most extreme circumstances (Loeve et al. 2001b). Hence, ferry flights will not be used in this research.

The following two *special restrictions* are also imposed:

(1) If an aircraft is en route to the reduced MOG station at time T_S, it will be allowed to land as originally scheduled, regardless of any reduced MOG violation. Only flights of relatively long duration (i.e., transcontinental or transoceanic flights) will cause this type of exception.

(2) Aircraft on the ground at the reduced MOG station at time T_S will be allowed to stay until their next scheduled departure, regardless of any reduced MOG violation.

It is assumed that the other airports (except for the reduced MOG station) always have sufficient capacity for the aircraft forced to stay due to the delayed and canceled flights.

In summary, the problem discussed in this chapter can be stated as: Given the original flight schedule, the MOG of one station is reduced for a known time period. This will require rescheduling the flights affected by the disruption. No ferry flights are allowed. The goal is to minimize the cost incurred by the changes to the schedule and insure return to the original schedule by the end of recovery window.

Let us consider the small illustrative problem example (altered from Bard et al. 2001), detailed in Table 4.1 and Figure 4.1, where three aircraft cover 12 flights.

Flight Number	Origin	Destination	Departure Time	Arrival Time	Flight Minutes	Cancellation Cost	Aircraft Assigned
0	BOI	SEA	14:10	15:20	70	7350	1
1	SEA	GEG	16:05	17:00	55	10231	1
2	GEG	PDX	17:40	18:40	60	7434	1
3	PDX	BOI	19:20	20:35	75	14191	1
4	SEA	BOI	15:45	17:00	75	11189	2
5	BOI	SEA	17:40	18:50	70	12985	2
6	SEA	GEG	19:30	20:30	60	11491	2
7	GEG	SEA	21:15	22:15	60	9581	2
8	GEG	PDX	15:15	16:20	65	9996	3
9	PDX	GEG	17:30	18:30	60	15180	3
10	GEG	PDX	19:10	20:20	70	17375	3
11	PDX	GEG	21:00	21:55	55	15624	3

Table 4.1: A Small Schedule

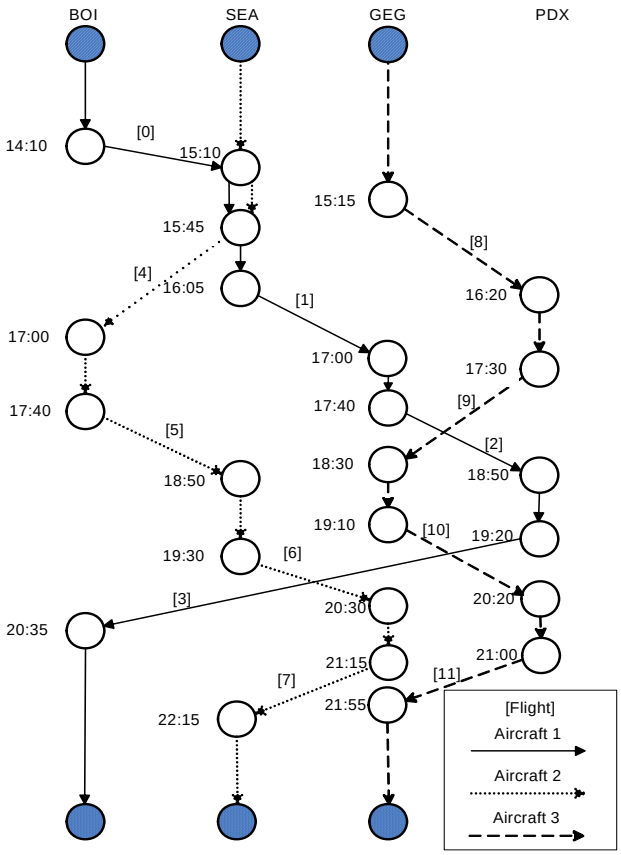

Figure 4.1: Time-Space Network of the Small Schedule

Suppose, that at 18:00 hours, we learn that the capacity of station PDX is to be reduced to zero from 19:30 to 21:30. Rescheduling some flights is required. Suppose the recovery window ends at 23:55. Two possible corrective actions are (1) to cancel flight 10 (which implies that flight 11 will also be cancelled) or (2) to delay the departure of flights 10 and 11 untill 20:20 and 22:10, respectively. Both solutions allow the schedule to resume originally scheduled operations by the end of recovery window 23:55.

4.2 METHODOLOGY

In this research, we consider five approaches for the reduced station MOG problem:

(1) a Pure TS approach (RSC-TS),

(2) a time-space network model (NM),

(3) a hybrid method (HM) combining NM with a Limited TS (LTS),

(4) HM+TS, HM followed by RSC-TS, and

(5) a local search method (LS) similar to GRASP (Argüello et al. 1997).

4.2.1 A Pure TS to the RSC, RSC-TS

4.2.1.1 The Objective Function

The RSC problem's objective function is the sum of the cost associated with delays, cancellations, and swaps plus Lagrangian penalties assessed for constraint violations associated with exceeding any capacity constraints or recovery time windows.

The delay cost is a function of delay minutes like in Section 3.1.1. The cancellation cost is uniquely predefined for each flight. The swap cost is a function of swap numbers as defined in Section 3.1.1. The two penalty costs in the objective function are quantified as follows: (1) The *cumulative* MOG violations are penalized based on the

number of aircraft in excess of M at each 15-minute check point within the reduced MOG time period $[T_s, T_e]$. The MOG violation penalty is equal to the product, *MOG Penalty Coefficient · sum of violations*. (2) If any route fails to return to the published schedule by more than 15 minutes later than T_E, a penalty term (*Recovery Window Penalty Coefficient · Number of route lateness violations*) is added to the objective function. The 15 minute extension is allowed because delays less than 15 minutes are not reported to FAA and can be easily corrected in downstream flights.

Thus the objective function is defined as

Objective = $C_d + C_c + C_s + P_1 + P_2$

where C_d: Total delay costs

C_c: Total cancellation costs

C_s: Total swap costs

P_1: Total Penalty for failing to satisfy the MOG constraint

P_2: Total penalty for exceeding the recovery time window by more than 15 minutes.

4.2.1.2 Constraints

Feasible solutions will satisfy the following constraints:

(1) Each aircraft departs from its latest arrival station.

(2) The interval between an aircraft arrival and its next departure can not be less than the minimum turnaround time.

(3) During the reduced MOG time period $[T_s, T_e]$, no more than M aircraft may be present at the reduced MOG station S.

(4) All flights originally scheduled to depart after the end of recovery window, T_E, must be able to fly as originally scheduled.

Requirements (1) and (2) are hard constraints, while (3) and (4) are included as Lagrangian penalty terms in the objective function.

As described in Section 4.2.1.1, the reduced MOG period [T_s, T_e] is discretized creating $i = 1, \ldots, k$ fifteen-minute check points where N_i is the associated number of aircraft on the ground at check point i. Two kinds of MOG violation penalties were investigated: a linear penalty

$$Penalty = \left[\sum_{i=1}^{k} \max(N_i - M, 0) \right] \cdot PC$$

and a quadratic penalty

$$Penalty = \left[\sum_{i=1}^{k} \max(N_i - M, 0) \right]^2 \cdot PC .$$

The penalty coefficient, *PC,* is a user defined value which must be set to a relatively large value to force the search to attempt to reduce the reduced MOG violations. Because it is preferred that no MOG violations be present, the penalty should be large enough to discourage MOG violations from existing.

However, from a practical perspective, small violations of the MOG restriction for short periods of time can be present in a revised schedule without incurring any penalty since they can be managed using standard operational techniques. For example, a single aircraft causing a MOG violation for a period of 10 to 15 minutes is not considered a *critical MOG violation* because the gate personnel and the non-flying crews of the affected aircraft can take action to compress the flight turnaround times on the gate resident aircraft and the gate awaiting aircraft to cause minimal disruptions in the resultant schedule (Bailey 2007).

The discretization of 15 minutes described above will disallow any MOG violation in excess of 14 minutes duration and no penalty will be assessed for non-critical violations of 14 minutes or less.

4.2.1.3 The RSC-TS Neighborhood Definitions

The RSC-TS neighborhood definitions, reminiscent of the AGP-TS definitions in Chapter 3, are:

1) Swap flights between two routes;
2) Insert flights between two routes;
3) Delay a flight - A flight may be delayed without changing the sequence of the flights in the route. Delay moves may be assigned in 15 minute increments up to a user-defined maximum delay;
4) Move a flight departure earlier - (in 15 minute increments *but* no earlier than its original scheduled departure time). This may also allow subsequent flights on the route to be moved earlier as well;
5) Cancel flights – A flight sequence which starts and ends at the same station may be cancelled;
6) Exchange route tails - When the two routes involved in the move (insert or swap) have different ending stations, their route tails (the flight series of the route after the end of recovery time window T_E) can be exchanged. This move is integrated as part of insert or swap move structures.

4.2.1.4 Construction of Recovery Windows for All Routes

Unlike the AGP in Chapter 3, the end of recovery window T_E in the RSC is not necessarily the end of the operation day. It is preferred that flights scheduled to depart after T_E depart no more than 15 minutes after the originally scheduled departure time.

This requires, for the *first flight* in a unique sub-route starting at any station departing after T_E, that a plane suitable for that flight arrive at the associated station at least 15 minutes prior to the scheduled departure time (presuming the minimum turnaround time is 30 minutes). This restricts the end result of the search associated with decisions affecting flights before T_E. If the search schedules a sub-route's first flight (departing after T_E) to depart more than 15 minutes after the originally scheduled departure time, an objective function penalty is charged.

4.2.1.5 Flow Chart

Figure 4.2 displays the flow chart of the RSC-TS algorithm. First the recovery window flights, the flights that may be rescheduled, are identified. The original flight schedule is used as the initial starting solution.

At the beginning of each iteration, the tabu memory structure determines if a specified number of solutions has been repeatedly visited recently. If this is true, the search is trapped into an attractor basin, causing the tabu memory structure to be cleared and invoking an escape process.

Depending on the current MOG violations, different neighborhoods are investigated according to a strategic dynamic neighborhood selection procedure. At each iteration, if MOG violations remain, the cancel flights and delay neighborhoods are investigated; otherwise, the insert, swap, delay and move earlier neighborhoods are investigated.

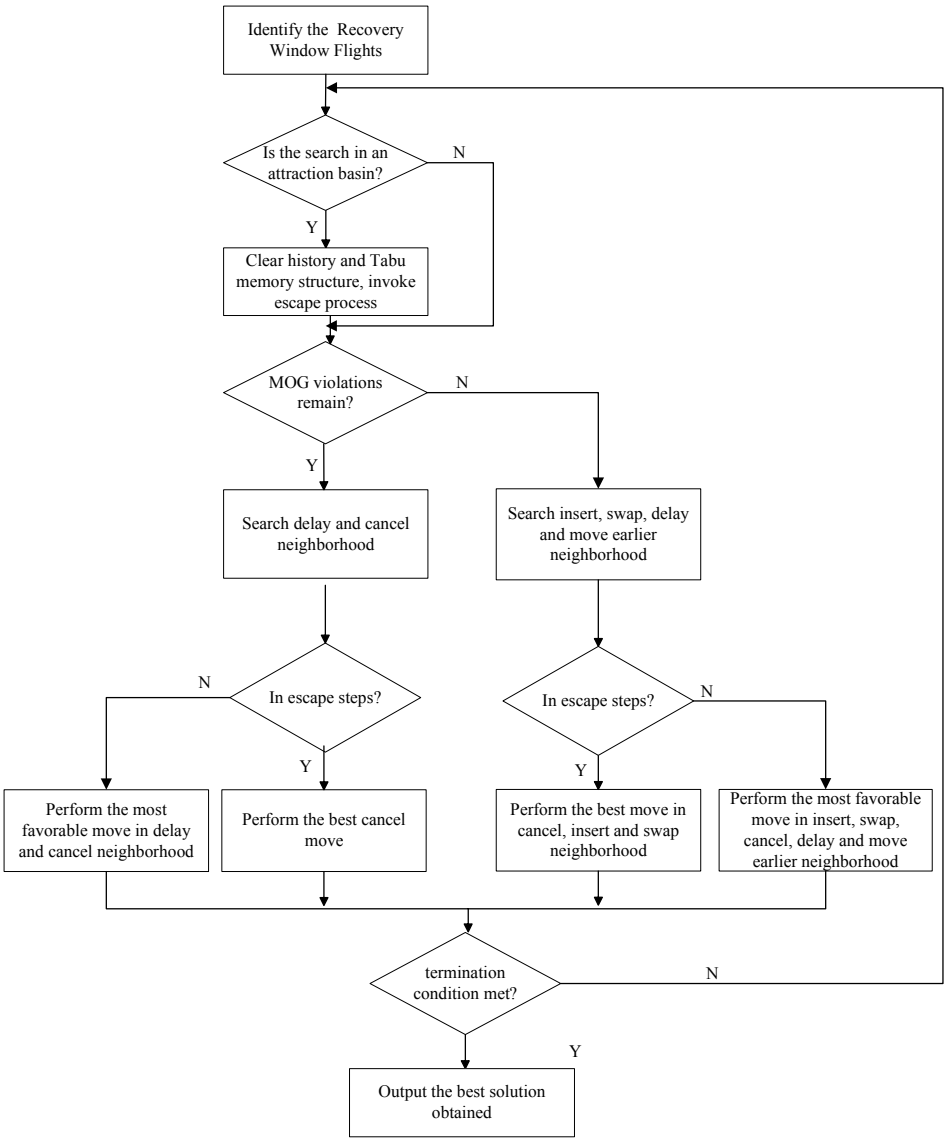

Figure 4.2: Flow Chart of TS-RSC

If MOG violations remain, and the search is in the escape mode, the best move in the cancel flights neighborhood is performed. Otherwise, the most favorable move from the delay or cancel flights neighborhood is performed.

If MOG violations do not remain, and the search is in the escape mode, the best move from the cancel flights, insert or swap neighborhoods is performed. Otherwise, the

most favorable move from the insert, swap, cancel flights, delay or move earlier neighborhood is performed.

The RTS methodology applied to the RSC problem is identical to the AGP-TS method discussed in Section 3.2.1. The search ends when the termination condition is met, i.e., the maximum time has elapsed.

4.2.2 HM - A Hybrid of NM and LTS

Similar to HM1 for AGP discussed in Chapter 3, HM attacks the RSC in two steps: first, a time-space network flow model is created with time discretization. The objective is to minimize the sum of delay cost and cancellation cost, without considering the swap cost. Again, the aircraft assignment information or the route information are not explicitly considered. The network flow model is solved by the CPLEX MIP solver. The arc-based solution is then post-processed to identify the resulting flight times from the arc values. A route-based initial solution is then generated by a one-pass route-generation procedure. LTS then uses this initial solution as the starting solution to diminish the swaps and find a good route-based solution. LTS is *limited* because it only modifies the aircraft assignment for flights while the flight schedule is unchanged.

Figure 4.3 presents an overview flow diagram of the HM methodology. We will now consider Figure 4.3 in more detail.

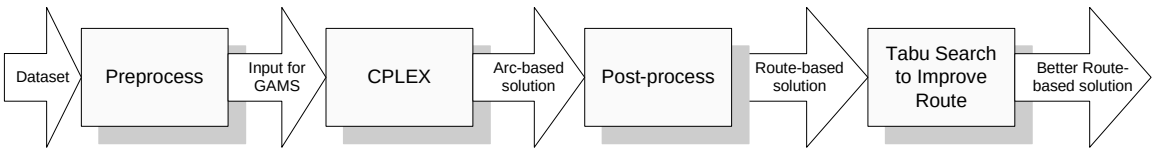

Figure 4.3: RSC-HM Flow Diagram

4.2.2.1 NM - Prohibiting All MOG Violations

The RSC problem, with a linear MOG violation penalty function, can be modeled as a time-space network flow problem with side constraints, NM. NM is similar to the model for the AGP, stated in Section 3.2.2.1. In the network, each node is associated with one station and a time point. The time points are generated by assigning different delay options for each flight. There are 18 different delay options (Thengvall et al. 2000) (in minutes): {0, 10, 20, 30, 40, 50, 60, 70, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360}.

Using CPLEX to attempt to solve NM with a quadratic MOG penalty failed due to the nonlinear term in the objective function. This is one of the limitations associated with the time-space network approach.

There are two kinds of arcs: grounding arcs start and end at the nodes of the same station indicating the aircraft remaining at the station during the time period; and flight arcs possess start and end nodes at different stations. All arcs are directed downward consistent with the orientation of the time axis. For the flight arcs, the start node location indicates the flight departure time, and the end node indicates the arrival time plus the minimum turnaround time, i.e., the next earliest possible departure time for the aircraft. Similar to the AGP described in Section 3.1, flight arcs incur associated delay costs and grounding arcs have zero arc costs. The cancellation cost of a flight, f, not explicitly included in arc cost, is associated with a binary variable which has value 1 if all flight arcs associated with flight f have value 0.

NM is different from the analogous AGP model in two ways: (1) Since the recovery window is not from the start of the operation day to the end of the operation day, the recovery window flights must be identified and (2) NM must impose the new reduced station capacity constraint which adds complexity to the model by requiring a set of side constraints on the total flow present on the grounding arcs and the arriving flight

arcs at any point in time during the reduced capacity period. The reduced station capacity constraint is included in the objective as a weighted Lagrangian penalty term. Specifying the penalty as a linear function of the MOG violations in the time-space network model is a simplification of the *actual* cost function. According to professionals from the airline scheduling industry, the MOG violation cost function would be a nonlinear function that grows (perhaps exponentially) with the number of excess aircraft on the ground. As noted earlier, even a quadratic MOG violation function causes CPLEX to fail. Unlike RSC-TS, NM will detect and penalize any occurrence of a MOG violation including non-critical MOG violations.

The network contains the nodes and arcs associated with flights departing between the start of the operation day and the end of the operation day. However, *only* recovery window flights may have the associated 17 nonzero-delay flight arcs described above. Seventeen nonzero-delay flight arcs is a suitable compromise in the number of such arcs included in the model to provide sufficient relative accuracy. Computational experiments have confirmed that network models with only 4 times as many nonzero-delay arcs will yield unacceptable long execution times for the practical application of the results to the routing schedule recovery. Flights before the recovery window have only the zero-delay flight arcs. Experimental studies have shown that allowing minor violations of the recovery time window may make it possible to obtain superior solutions. Considering that delays less than 15 minutes are not reported to FAA and should be easily corrected in the downstream flights, the RSC model allows the flights originally departing after recovery window flights to delay departure for as much as 10 minutes, i.e., such flights are allocated 10-minute delay flight arcs (since a 15 minute delay is not included in the 17 nonzero-delays in the network model).

The network is generated as follows: All flights in the schedule are used as the base to generate the network by creating the associated zero-delay flight arcs. For each station, the station supply node (like the shaded nodes at the top of Figure 4.1) possesses an input flow into the network equal to the number of aircraft present at the station at the start of the operation day. The station demand node (like the shaded nodes at the bottom of Figure 4.1) absorbs arriving flights to this station at the end of the operation day. The intermediate nodes allow departures from the origin stations at specific discrete time points and arrivals at associated destination stations for each different delay option for each flight. Next, for each route, the recovery window flights are selected. An appropriate set of flight arcs are created for each recovery window flight consistent with the various non-zero delay options. (Those flights governed by the special restrictions, discussed in Section 4.1, are excluded.) A single 10-minute delay arc is created for each of the flights departing after T_E. Finally, grounding arcs are created by connecting each of two adjacent nodes belonging to the same station downward through time.

In the example shown in Figure 4.1, at T_S = 18:00 hours, we learn that the capacity of station PDX is to be reduced to zero from T_s =19:30 to T_e = 21:30 and the recovery window ends at T_E = 23:55. The network flow model includes the nodes and the arcs associated with flights departing between the start of the day and T_E, which in this case, are all of the flights shown in Figure 4.1. The recovery window flights are 3, 6, 7, 10 and 11 and, therefore, are the only flights that may be cancelled or have their departure times modified. Only flights 3, 10 and 11 are directly associated with station PDX. However, in general, the other recovery window flights may need to be modified to achieve superior solutions. For example, it may provide more options for modifying the routes and aircraft assignments. Thus only these 5 flights have non-zero-delay arcs.

The reduced station capacity constraint is enforced by restricting the total flow entering any member of the set of nodes which is associated with the restricted station during the reduced MOG time period. However, just as in the network flow model of Chapter 3, the time associated with the destination node of a flight arc is actually the flight arrival time plus the minimum turnaround time. For example, suppose we would like to count the number of aircraft at station *S* at 10:30 hours. Let *n* be the node associated with station *S* at time 10:30. Counting the flows ending at node *n* is incomplete because there may be situations where a flight arc ends at another node associated with station *S* at 10:45. Because, in the network model, the flight arc end node is associated with actual arrival time plus minimum turnaround time, (which is 30 minutes in 737 dataset) this aircraft *actually* arrived at station *S* at 10:15 and stayed at station *S* until at least 10:45. Hence, we must also count the flow on this flight arc. Therefore, in general, if node *n* is associated with station *S* and time *t*, the number of aircraft present should be the sum of: Flows entering node *n* plus flight flows entering any node *m* associated with station *S* that has a time that falls strictly in the open interval, (*t, t + minimum turnaround time*).

For all time points within the reduced MOG time period, the number of aircraft at station *S* is determined by the above method. The reduced MOG violations are then summed, multiplied by the Lagrangian coefficient and added into the objective function.

The mathematical model for NM can be formulated as follows:

Indices:

n: nodes

a: arcs

f: flights

Sets:

N: set of nodes, $n \in N$ where each node has an associated station, s_n and time, t_n

S: set of supply nodes

D: set of demand nodes

A: set of arcs

A': set of arcs associated with recovery window flights

F: set of flights

F': set of recovery window flights

R: set of nodes associated with both the reduced MOG time period and the reduced MOG station

I(n): set of arcs entering node *n*

O(n): set of arcs leaving node *n*

G*(f)*: set of flight arcs associated with flight *f*

E(*n*): set of the nodes associated with node n and its station s_n whose times are strictly in the open interval,

(t_n , t_n + minimum turnaround time)

Parameters:

C_a: cost of arc *a* per unit flow

P_a: type of arc *a*

σ_n: flow supply at node n

δ_n: flow demand at node n

β_f: cancellation cost of flight f

α: penalty (per aircraft) of exceeding reduced MOG

M: capacity of reduced MOG station during reduced MOG time period

Variables:

x_a: amount of flow on arc a

y_f: cancellation indicator for flight f

z_n: number of aircraft exceeding the reduced MOG capacity at node $n \in R$

Mathematical Formulation:

$$\text{Minimize} \sum_{a\in A',P(a)=flight} C_a x_a + \sum_{f\in F'} \beta_f y_f + \sum_{n\in R} \alpha z_n$$

Subject to:

(aircraft balance at supply node) $$\sum_{a\in O(n)} x_a = \sigma_n, \forall n \in S \quad (1)$$

(aircraft balance at demand node) $$\sum_{a\in I(n)} x_a = \delta_n, \forall n \in D \quad (2)$$

(aircraft balance at intermediate node) $$\sum_{a\in I(n)} x_a = \sum_{a\in O(n)} x_a, \forall n \in N \setminus (D \bigcup S) \quad (3)$$

(flight coverage) $$\sum_{a\in G(f)} x_a + y_f = 1, \forall f \in F' \quad (4a)$$

$$\sum_{a\in G(f)} x_a = 1, \forall f \in F \setminus F' \quad (4b)$$

(MOG constraint) $$z_n \geq \sum_{a\in I(n)} x_a + \sum_{a\in K(n),P(a)=flight} x_a - M, \forall n \in R \quad (5)$$

$$where\ K(n) \equiv \bigcup_{m\in E(n)} I(m)$$

(binary flight cancellation) $$y_f \in \{0,1\}, \forall f \in F' \quad (6)$$

(arc flow) $$x_a \in Z_+ = \{0,1,...\}, \forall a \in A \quad (7)$$

(number of aircraft on ground exceeding reduced MOG capacity) $$z_n \geq 0, \forall n \in R. \quad (8)$$

The objective function is the sum of cancellation cost, delay cost and the Lagrangian penalty to the MOG violation, which is to be minimized. Constraints (1), (2) and (3) state the flow balance at each node. Flight coverage is ensured by constraints (4a) and (4b). Constraint (5) enforces the reduced station capacity constraint, where the right hand side is composed of three parts. The first part is the sum of flows entering the node *n*. As explained above in this section, the second part is the sum of flight arcs entering the nodes which are associated with the same station as node *n* but have times later than node

n but within the minimum turnaround time. The third term is the reduced station capacity, *M*.

The NM model is NP-hard because it is a network flow model with side constraints (4) and (5) (Garey et al. 1979). Experiments to solve this model were performed with the CPLEX solver.

4.2.2.2 NM - Allowing Non-critical MOG Violations

To allow non-critical MOG violations, the NM model presented in Section 4.2.2.1 is modified so that MOG violations are examined only at specified time check points 15 minutes apart.

The model is modified as follows:

(MOG constraint) $$z_n \geq \sum_{a \in I(n)} x_a + \sum_{a \in K(n), P(a)=flight} x_a - M, \forall n \in R' \qquad (5')$$

where $K(n) \equiv \bigcup_{m \in E(n)} I(m)$.

The difference between the new constraint (5') and the previous constraint (5) is that it only applies to the nodes in the set of *R'* which are equally spaced on the time axis 15 minutes apart starting at the beginning of the reduced MOG time period.

NM uses the same route-generation technique that was employed in the AGP detailed in Section 3.2.2.2 and the LTS method to find the best routes as discussed in Section 3.2.2.3 where 100 iterations are allowed.

The detailed computational results of NM and HM allowing non-critical MOG violations are presented in Section 4.3.

4.2.3 HM Followed By TS - HM+TS

In HM+TS, the HM result is used as initial starting solution for RSC-TS in the hope that an improved solution can be achieved. The RSC-TS method used in isolation

terminated in 30 seconds of computational time. When RSC-TS is part of HM+TS, it is terminated after performing 100 iterations. The computational results are described in Section 4.3.

4.2.4 Local Search - LS

LS, derived from the GRASP approach of Argüello et al. (1997), was also implemented for the RSC problem. It used the same move neighborhood as RSC-TS and employed a restricted candidate list (RCL) to store the elite solutions discovered in the search. In each iteration, one solution from the RCL is randomly selected as the incumbent solution for the neighborhood search. Figure 4.4 presents the pseudocode of the local search algorithm.

```
 incumbent = the initial schedule
 best_solution = incumbent
While (stop criteria not met)
{ If ( there are MOG exceeding )
    { Set the neighborhood to {delay, cancel}
    }
 else
 { Set the neighborhood to {insert, swap, delay, move_earlier }
 }
 While (searching neighborhood of the incumbent has not finished)
 { sol = neighbor of incumbent
  If ( sol meets the requirement of RCL ) { put sol in RCL }
  If ( obj. of sol < obj. of best_solution )  { best_solution = sol  }
 }
 Randomly chose one solution from RCL as incumbent
}
```

Figure 4.4: Pseudocode of Local Search for RSC

In the LS experiments, a maximum of 330 seconds running time was allowed. The candidate list consisted of no more than 50 solutions. The newly found neighborhood solution was put in the RCL if the objective function value is less than 1.1·(incumbent objective function value). When the RCL was full, the least desirable solution is replaced by a new better solution.

4.3 COMPUTATIONAL EXPERIMENTS

4.3.1 The Experimental Dataset

The RSC experiments were performed using the 737 dataset (Argüello 1997) described in Chapter 3 which embodied a hub-spoke schedule with station EWR (Newark International Airport) as the hub. It comprises 162 flights covering one operation day by 27 aircraft among 30 airports. A thirty minute turnaround time was stipulated between the arrival time and the subsequent departure in the aircraft route.

In all problem instances, the reduced MOG station was EWR. The capacity reduction starts at 10AM for all problem instances. The reduced MOG time periods were 2 hours, 4 hour, 8 hours and 12 hours respectively. As detailed in Table 4.2, the recovery time windows were stipulated according to suggestions from airline professionals.

Restricted Time period	Recovery Window Starts	Recovery Window Ends
12 hours	6 hours before	6 hours after
8 hours	4 hours before	4 hours after
4 hours	3 hours before	3 hours after
2 hours	2 hours before	2 hours after

Table 4.2: Recovery Window for RSC Problem

Based on the current undisrupted schedule, the nominal available MOG at EWR for this fleet of 27 aircraft is specified to be 7 aircraft. The problem instances were generated by restricting the capacity to 1, 3, 4 or 5 for the above reduced MOG time

periods. Table 4.2 gives the time frame and reduced capacity of the 14 problem instances that were generated and solved. Table 4.3 gives the disruption information for each of the problem instances. The meaning of each column is as follows:

- *Number of Affected Flights*: number of flights which land at EWR or leave EWR during the reduced station capacity period.
- *Number of Affected Routes*: number of routes which contain the affected flights.
- *Maximum Possible MOG Overage*: $\sum_i \max(N_i - M, 0)$, where i indexes the check points (15 minutes apart) when the number of aircraft at EWR are counted throughout the reduced station capacity period, N_i is the number of aircraft at EWR at check point i, and M is the reduced station capacity.

Problem Instance	Reduced Capacity Duration	Capacity Reduced to	Reduced Capacity Start Time	Reduced Capacity End Time	Recovery Window Start Time	Recovery Window End Time	Number of Affected Flights	Number of Affected Routes	Maximum Possible MOG Overage
1	12hr	5/7	10:00	22:00	4:00	4:00 (2nd day)	66	19	6
2	12hr	4/7	10:00	22:00	4:00	4:00 (2nd day)	66	19	17
3	12hr	3/7	10:00	22:00	4:00	4:00 (2nd day)	66	19	32
4	12hr	1/7	10:00	22:00	4:00	4:00 (2nd day)	66	19	87
5	8hr	5/7	10:00	18:00	6:00	22:00	45	16	1
6	8hr	4/7	10:00	18:00	6:00	22:00	45	16	7
7	8hr	3/7	10:00	18:00	6:00	22:00	45	16	15
8	8hr	1/7	10:00	18:00	6:00	22:00	45	16	48
9	4hr	4/7	10:00	14:00	7:00	17:00	19	12	3
10	4hr	3/7	10:00	14:00	7:00	17:00	19	12	6
11	4hr	1/7	10:00	14:00	7:00	17:00	19	12	20
12	2hr	4/7	10:00	12:00	8:00	14:00	9	8	3
13	2hr	3/7	10:00	12:00	8:00	14:00	9	8	6
14	2hr	1/7	10:00	12:00	8:00	14:00	9	8	14

Table 4.3: Disruption Information for RSC Problem Instances

RSC-TS, HM, HM+TS and the local search were applied to each of the 14 problems. For HM and HM+TS, the network flow model allowing non-critical MOG violations is used. In both uses of RSC-TS for the results reported in Table 4.6, the linear

objective function was employed. The remaining part of the chapter presents and discusses the results obtained.

4.3.2 Results from RSC-TS

Table 4.4 presents the best solution results obtained by RSC-TS with linear MOG violation penalty (indicated as TS_linear) and TS with quadratic MOG violation penalty (indicated as TS_quadratic) for each problem. The MOG penalty coefficient (PC) is set to be 50000 for TS_linear, and 16000 for TS_quadratic. These coefficients were selected so that they are large enough to discourage critical MOG violations. The bold font indicates the superior value obtained. If the values are identical then only a single number is presented.

The meanings of each column are as follows:

Number of Delayed Flights: how many flights are delayed.

Number of Cancelled Flights: how many flights are cancelled.

Sum of Delayed Minutes: sum of delay minutes for all delayed flights.

Number of Swaps: number of route swaps (compared to the original routes).

Number of Intact Routes: number of un-altered routes (their flight sequences are the same as the original routes).

For all of the 14 problem instances, the best solution removed all critical MOG violations for both TS methods, so the MOG penalty terms in their objective functions are of zero. Therefore, though they are associated with different MOG violation penalty functions, we may still properly compare the two objective function values. TS_quadratic obtained better solutions than TS_linear for problems 3, 4 and 11; TS_linear obtained slightly better solutions than TS_quadratic for problem 8. For the other 10 problems, identical solutions were obtained.

Problem Instance	Number of Delayed Flights	Number of Canceled Flights	Sum of Delayed Minutes	Number of Swaps	Objective Function Value (Linear/Quadratic)	Number of Intact Routes	Best Solution Found at Iteration	Time Used at Finding Best Solution (s)
1	7	0	222	4	4,298	25/25	4	3/4
2	18	0	787	5	15,498	24/24	12	15/16
3	32/29	2	2345/2259	14/17	53,058/**52,371**	19/18	23/24	30/30
4	43/42	16/18	7725/6913	14/15	194,193/**182,395**	19/18	40/42	29/30
5	1	0	15	0	300	27/27	1	1
6	5	0	114	4	1,998	25/25	4	2
7	15	0	285	6	4,940	24/24	23	20
8	32/11	20/22	2190/1632	11/18	**88,809**/89,370	19/18	69/54	30
9	3	0	68	2	964	25/25	2	1
10	6	0	164	2	2,884	25/25	3	1
11	18/12	4/6	846/563	5/11	26,688/**26,134**	24/20	73/36	18/8
12	2	0	40	0	610	27/27	1	1
13	4	0	80	0	1,220	27/27	2	1
14	11	0	265	4	4,918	25/25	8	1

Table 4.4: Best Result Obtained by RSC-TS (Linear/Quadratic)

Table 4.4 also provides information on the iterations and the time used to find the best solution where a maximum of 30 seconds was allowed. For those 10 problems where the two algorithms obtained the same result, the time and iteration numbers used are about the same for the algorithms. TS_quadratic obtained the better solution in a shorter time for problem 11, obtained a better solutions in about the same time for problem 3 and 4, and obtained a worse solution in the same time for problem 8.

Apparently for the problems where different solutions were obtained, the different MOG violation penalty functions diverted the trajectory of the search during the execution of the RSC-TS.

4.3.3 Comparing the Network Model (NM) and HM

Table 4.5 presents the results from NM and HM. Using LTS in HM requires only 1 to 3 additional seconds to greatly diminish the swap costs, *dramatically* improving the objective function values for all 14 problems.

Problem Instance	Number of Delays	Number of Cancellations	Sum of Delay Minutes	Sum of Cancellation cost and Delay Cost	Number of Swaps	Number of Intact Routes	Objective Function Values (Linear/Quadratic)	Time Used at Finding Best Solution (s)
1	6	0	120	2,020	49/2	0/25	28,695/**2,064**	48/51
2	14	0	310	5,630	50/8	1/21	33,405/**6,341**	49/50
3	26	0	630	11,080	52/8	0/21	41,121/**11,791**	48/50
4	40	20	1890	82,190	53/22	0/11	113,398/**87,567**	49/50
5	1	0	10	10	52/0	0/27	30,051/**10**	44/45
6	5	0	110	2,010	50/2	1/25	29,785/**2,054**	44/45
7	15	0	280	4,270	53/4	0/23	35,478/**4,448**	46/47
8	28	18	680	52,900	49/18	0/11	79,575/**56,500**	46/47
9	4	0	90	1,600	50/0	1/27	29,375/**1,600**	25/26
10	7	0	170	3,200	52/2	1/25	33,241/**3,244**	23/24
11	11	6	250	18,020	47/10	1/19	42,562/**19,131**	23/24
12	5	0	70	610	50/0	1/27	28,385/**610**	22/23
13	8	0	120	1,220	52/0	1/27	31,261/**1,220**	11/12
14	9	6	150	15,810	49/4	1/23	42,485/**15,988**	11/12

Table 4.5: Comparison of the Result from NM/HM

4.3.4 Comparing RSC-TS, NM, HM, and HM+TS

Table 4.6 presents the objective function values and the times used for the best solution obtained by the following methods: RSC-TS, NM, HM and HM+TS. For all of 14 problems, the best results from all algorithms have no critical MOG penalties present. The best results (including ties) are indicated by **bold** font, except in problems 5, 11, 12, and 13 where the *italicized* results from HM+TS did not improve the best result found by HM.

Problem Instance	Objective Function Values (RSC-TS/NM/HM/HM+TS)	Time Used at Finding Best Solution (s)
1	4,298/28,695/2,064/**1,674**	3/48/51/54
2	15,498/33,405/6,341/**5,250**	15/49/50/53
3	53,058/41,121/11,791/**11,130**	30/48/50/109
4	194,193/113,398/87,567/**85,760**	29/49/50/55
5	300/30,051/**10**/*10*	1/44/45/46
6	**1,998**/29,785/2,054/2,040	2/44/45/46
7	4,940/35,478/4,448/**4,230**	20/46/47/63
8	88,809/79,575/56,500/**55,604**	30/46/47/55
9	**964**/29,375/1,600/*1,600*	1/25/26/27
10	**2,884**/33,241/3,244/3,184	1/23/24/39
11	26,688/42,562/**19,131**/*19,131*	18/23/24/25
12	**610**/28,385/**610**/*610*	1/22/23/24
13	**1,220**/31,261/**1,220**/*1,220*	1/11/12/13
14	**4,918**/42,485/15,988/*15,988*	1/11/12/13
Number Superior	6/0/4/6	14/0/0/0

Table 4.6: Comparison of Objective Function Values for RSC Problem from Different Methodologies: RSC-TS/NM/HM/HM+TS

RSC-TS obtained the unique best result in 4 of the 14 problems and shared in the best result with HM in 2 problems. HM obtained the best result in 4 of the 14 problems. HM+TS obtained the unique best result in 6 of the 14 problems and improved the HM results on 8 problems.

RSC-TS took significantly less time than the other methodologies. For most of the problems (except for problem 3), HM+TS took only a little more time than HM.

HM, in stage 1, optimally solves the network flow model minimizing the sum of delay costs and cancellation costs; in stage 2, the LTS *greatly* lessens swap costs.

When comparing to HM, RSC-TS:

(1) Treats the sum of delay cost, cancellation cost and swap cost as an integrated objective function;

(2) Unlike HM which allows only specified discrete delay time periods, TS allows any delay length (expressed in minutes) up to its allowed maximum time period. This capability is derived from the dynamic neighborhood search methodology.

Problem 14 has the shortest MOG period (2 hours) with the smallest reduced MOG capacity (1 aircraft) among all of the problems. RSC-TS provides a *dominantly* superior solution (with no canceled flights) with an objective function value of 4918 (31% of the best HM solution's objective function value). HM can not obtain the RSC-TS solution because of the presence of delays of 18 and 22 minutes, *not* permitted in HM. HM's best solution has 6 cancelled flights and an objective function value of 15998.

HM experiments were also conducted with 73 five-minute delay intervals (as opposed to the 18 delay intervals described in Section 4.2.2.1). The required computational effort increased over tenfold and the results were only marginally improved.

It is interesting to note that experiments starting RSC-TS from the solution yielded by HM, where ***all*** *MOG violations were prohibited*, yielded quite superior *results* for problems 6 (*1430*), 7 (*3854*) and 10 (*2298*) when compared to the best results in Table 4.6. These improvements were likely made possible by more flexible starting solutions provided when no MOG violations were allowed. The results of these experiments are presented in Appendix D.

In ***all*** 14 problems, some form of tabu search, either LTS or RSC-TS, was used to obtain the best solution.

4.3.5 Compare RSC-TS and LS

Table 4.7 presents the comparative results between Local Search (LS) and TS, with linear and quadratic MOG penalty function (denoted as LS_linear and LS_quadratic,

respectively) for the 14 problems. The MOG penalty coefficients used are 100000 in LS_Linear and 32000 in LS_quadratic. These coefficients were experimentally obtained and yielded the best overall result.

The last row in the table presents the number of problems in which each algorithm obtained the best objective function value. The LS and TS algorithms obtained the same result for 2 of the problems. For 10 problems, TS obtained better total objective function values than LS. Only for problem 3 did LS obtain a better solution than TS. However, for problem 3, the objective function value of TS result (52371) was comparable to that of local search (50640) while TS took significantly less time than local search did (30 seconds v.s. 307 seconds).

TS was allowed 30 seconds and LS was allowed 330 seconds. Seven LS times to find the best solution exceeded 200 seconds. Except for problems 5 and 12 in which the two methods took the same time, LS took significantly longer time than TS.

Hence, TS generally obtained better objective function values in a shorter time than LS.

Problem Instance	Number of Delays	Number of Cancellations	Sum of Delay Minutes	Number of Swaps	Number of Intact Routes	Objective Function Value RSC-TS (Linear/Quadratic) LS (Linear/Quadratic)	Time Used at Finding Best Solution (s)
1	**7**/**7**/10/10	0	**222**/**222**/557/557	**4**/**4**/6/6	**25**/**25**/24/24	**4,298**/**4,298**/11,540/11,540	**3**/4/129/129
2	**18**/**18**/28/28	0	**787**/**787**/2128/2128	**5**/**5**/11/11	**24**/**24**/22/22	**15,498**/**15,498**/43,754/43,754	**15**/16/320/321
3	32/**29**/32/32	2/2/**0**/**0**	2345/**2259**/2452/2452	14/17/**12**/**12**	19/18/**21**/**21**	53,058/52,371/**50,640**/**50,640**	**30**/**30**/308/307
4	43/**42**/67/67	16/18/**2**/**2**	7725/**6913**/15406/15406	14/15/**1**/**1**	19/18/**26**/**26**	194,193/**182,395**/314,221/314,221	**29**/30/297/290
5	1	0	15	0	27	300	1
6	**5**/**5**/14/14	0	**114**/**114**/579/456	**4**/**4**/13/9	**25**/**25**/20/22	**1,998**/**1,998**/12,958/9,390	**2**/**2**/266/294
7	**15**/**15**/29/21	**0**/**0**/**0**/2	**285**/**285**/2109/1719	6/6/12/**5**	**24**/**24**/21/**24**	**4,940**/**4,940**/43,590/38,908	**20**/**20**/299/273
8	32/**11**/36/33	20/22/**16**/**16**	2190/**1632**/8839/7271	**11**/18/14/12	**19**/18/21/**19**	**88,809**/89,370/219,123/184,602	**30**/**30**/288/289
9	3	0	68/68/**66**/69	**2**/**2**/4/4	25	**964**/**964**/1,098/1,098	**1**/**1**/161/228
10	**6**/**6**/7/7	0	**164**/**164**/172/182	**2**/**2**/5/5	**25**/**25**/24/24	**2,884**/**2,884**/3,278/3,688	**1**/**1**/289/173
11	18/**12**/19/15	**4**/6/10/6	846/**563**/997/1327	**5**/11/8/11	**24**/20/**24**/21	26,688/**26,134**/43,341/41,114	18/**8**/306/296
12	2	0	40	0	27	610	1
13	4	0	**80**/**80**/**80**/100	**0**/**0**/**0**/4	**27**/**27**/**27**/25	**1,220**/**1,220**/**1,220**/1,988	**1**/**1**/92/192
14	11/11/**8**/11	**0**/**0**/4/2	**265**/**265**/332/432	**4**/**4**/8/6	**25**/**25**/24/24	**4,918**/**4,918**/15,741/14,920	**1**/**1**/146/145
Number Superior	**9/13/5/4**	**11/10/12/11**	**9/13/4/2**	**11/9/5/5**	**12/10/7/7**	**11/12/4/3**	**13/11/1/1**

Table 4.7: Comparison of the Result from RSC-TS (Linear/Quadratic) and LS (Linear/Quadratic)

4.4 CONCLUSIONS

Prior to the efforts documented in this dissertation, the RSC Problem has not been addressed. In the research documented here, five methodologies were studied to attack this problem: (1) a TS method, (2) the NM method, (3) HM combining NM and LTS, (4) HM followed by TS (HM+TS), and (5) LS. In all studies documented in this chapter, the best solution found involved the use of tabu search, either in the form of LTS or RSC-TS.

In Chapter 5, conclusions and directions for future research will be discussed.

Chapter 5

Conclusions and Suggestions for Future Research

Since disruptions in the carefully constructed day-to-day schedules of commercial passenger aircraft *frequently* occur, effective and efficient disruption management techniques are becoming more and more important to the success of the airline industry. A timely recovery methodology yielding low operational costs with minimal deviations from the original plan is greatly preferred whenever a disruption occurs.

The research documented in this dissertation have shown conclusively that integrating TS with classical optimization methods provides great potential for improving the results of a disruption management technique. Indeed, in every example problem studied herein, TS did contribute to obtaining the best solution found.

5.1 SUMMARY OF CONTRIBUTIONS

Applying TS to the flight rescheduling problem has not been extensively investigated before. This research focused on using advanced tabu search techniques to solve two problems in airline disruption management problems, the aircraft grounding problem (AGP) and reduced station capacity problem (RSC). In both problems, various methodologies were investigated.

5.1.1 AGP Problem

One or more aircraft may be out of service for a period of time during the airline operation. Canceling the flights associated with the grounded aircraft will introduce additional costs and prohibit a recovery to the original schedule by the end of the preferred recovery window. Rescheduling the flights, including reassigning flights to

aircraft and choosing appropriate flights to cancel or delay, may decrease the additional costs caused by the disruption and allow the flight schedule to return to the published schedule by the end of the preferred recovery window with limited deviation from the original schedule.

Various methodologies using tabu search were developed and investigated to solve this problem: a pure TS (AGP-TS), a hybrid method (HM1) which was composed of a time-space network flow model and a limited TS, and another hybrid method which combined the first two methods (HM2).

In the computational experiments that were performed AGP-TS found comparable solutions to both hybrid methods in relatively short time. HM1 obtained solutions with superior objective function values to AGP-TS in most of the problem instances tested. Most problem solutions from HM1 were improved by HM2.

5.1.2 RSC Problem

For the first time, in this dissertation, the RSC problem is formally addressed where the MOG at the station during a specified time period is reduced. This dissertation focused on problems where reduced MOG for a hub station in a hub-spoke system was addressed. Similar to AGP problem, the goal was to minimize the associated cost and deviation from the original route and recover to the published schedule by the end of recovery window.

RSC was investigated using RSC-TS, NM, HM, HM+TS and LS. RSC-TS identified the number of aircraft during the reduced MOG time period and imposed a Lagrangian penalty for the violation of MOG constraint and revised the routes and the schedule to enforce this constraint. Experiments showed that RSC-TS usually found comparable solutions in short time. It outperformed LS and found 4 unique best solutions when compared to the remaining approaches.

NM used classical methods to partially solve the RSC problem while ignoring swap costs. HM combined NM and LTS and yielded solutions superior to RSC-TS for a subset of the problems studied. HM required significantly longer computational times than RSC-TS. HM+TS improved the HM result for 8 of the 14 problems studied.

In both the AGP and RSC problems, TS was easily capable of handling non-linear terms in the objective function. For the network model, in both problems, adding non-linear terms to the objective function caused CPLEX to fail. In the AGP problem, the swap cost is a quadratic function of swap numbers. The NM model can not model swap numbers. In the multicommodity network flow model formulated for the AGP, CPLEX was unable to solve problems with a non-linear term in the objective function.

In the RSC problem, MOG violations were penalized either by a linear or a non-linear function. Non-linear functions caused CPLEX to fail. RSC-TS encountered no difficulty with such nonlinear terms, utilizing similar amounts of computational effort as were used for a strictly linear objective function.

5.2 FUTURE WORK

Airline disruption management has always been a challenging and important problem requiring very timely solutions so that operations can be restored to the published flight schedule with minimal cost and passenger inconvenience. Practical scenarios can be much larger and more complicated than the models considered in this dissertation. There are many other interesting and challenging problems in this domain which should also be attacked using TS and hybrid methods such as those described above. Such problems include crew rescheduling and solving flight rescheduling and crew rescheduling problem as an integrated problem.

There are many extensions to the two problem addressed in this dissertation which also should to be investigated. Two of these involve (1) the addition of explicit

proactive stochastic considerations and (2) the dynamic superposition of a later disruption before the current disruption is resolved.

In this research, all disruptions are deterministic, i.e., the disruption scale and recovery time window are predefined known constants. It would be interesting to introduce practical stochastic elements into the disruptions, such as for weather-related disruptive events.

In real-life airline operations, one or more new disruptions could occur before the previous disruption was fully recovered. The current disruption management plan must be modified immediately to consider the new disruption. It would be interesting to investigate how well TS would function in such a dynamic problem domain.

Myriad other types of airline disruption management problems can be easily envisioned. These include simple multidimensional extension such as the multi-fleet flight rescheduling problem and the multi-station reduced capacity problem,

Finally, new more complex search neighborhoods for the use of TS as applied to the AGP and RSC problems should be investigated. Such investigations should include moves involving three or more routes. It would be interesting to determine if new neighborhoods could augment the performance of the current neighborhood definitions and possibly overcome the inferred feasible sub-regions of the solution space without explosively increasing the search complexity.

Attacking Disruption Management Problems by advanced tabu search methodology is an interesting topic. For other industries and applications, there surely are also many disruption management problems which may be approached by TS.

Appendix A

Pseudocode for TS-AGP Algorithm

A.1 CREATE INITIAL SOLUTION PSEUDOCODE

This pseudocode states how the initial starting solutions are generated in TS-AGP.

```
createStartSolution()
{  // First, collect all the information needed to construct all possible initial starting solutions
   initialize the array status[] to be UNPROCESSED
   // first round: cancel routes if it starts and ends at the same station
   for ( each grounded aircraft i )
   {  if ( status[ i ] == UNPROCESSED )
      { if ( route i starts and ends at the same station )
        {    move route i to cancellation route;   status[ i ] = MOVE_TO_CANCEL;
   } }   }
   // second round: check if can be canceled together with other route
   for ( each grounded aircraft i with status UNPROCESSED )
   {    for ( each grounded aircraft j with status UNPROCESSED )
        {  if ( i != j )
           {  if (route i starts/ends at the same station which route j ends/starts)
              {   // cancel them together
                  move route i and route j to cancellation route respectively
                  status[ i ] = MOVE_TO_CANCEL;    status[ j ] = MOVE_TO_CANCEL;
              }
              else if (end station of route i == start station of route j)
              { //  append route j to the end of route i
                status[ j ] = TAILED;   tailTo[ j ] = i;
              }
              else if (end station of route j == start station of route i)
              { //  append route i to the end of route j
                   status[ i ] = TAILED;
        tailTo[ i ] = j;
   } }    }
   // third round: for those not cancelled, find which are possible un-grounded routes to append to
   for ( each grounded aircraft i with status UNPROCESSED )
   {    // fills in the list appendCans[ i ], which is a list of route candidates to which route i may be appended to
        for ( each ungrounded aircraft k )
```

```
      { if (end station of route k == start station of route i)
          {   add k to appendCans[ i ]
              status[ i ] = FIXED;  //indicated it is settled
} }    }
 // fourth round: if there is no applicable route to append to, search in the grounded routes for appending
 for ( each grounded aircraft i with status UNPROCESSED )
 {   for ( each grounded aircraft j )
     { if ( i != j ) && ( status[j] != MOVE_TO_CANCEL ) && (end station of route j == start station of route i)
          {  append route of i to the end of j
             status[ i ] = TAILED;
             tailTo[ i ] = j;
 } }     }
 // after all the information for constructing initial starting solution has been collected, construct all possible initial
      starting solutions
 construct all initial starting solutions by permutation based on the value of the arrays appendCans[] and status[]
 select the initial starting solution with median objective function value
}
```

A.2 TABU SEARCH MAIN PSEUDOCODE

This pseudocode describes the Reactive Tabu Search in AGP-TS.

```
main()  /*  Reactive Tabu Search  */
{  ts_tenure = g_totalAircraftNum * INIT_TENURE;
   ts_tenure_max = max( ts_tenure*1.7 ,  ts_tenure + 5);   ts_tenure_min = max( min ( ts_tenure*0.5, ts_tenure - 5), 2 );
   bClearHistory = false;
   createStartSolution ();   //generate the initial starting solutions
   while ( search time limit is not reached)
   {   // check if the bClearHistory has been set by the last iteration
       if (bClearHistory )
       {   clear items which are marked "frequent" from g_mapSolutionHistory;   bClearHistory = false;
       }
       if ( bEscape )&&( nRemainingEscapeSteps > 0)
       {  nRemainingEscapeSteps --;
          if ( nRemainingEscapeSteps == 0 )
          {   bEscape = false;
       }   }
       //search the neighborhood for the moves
       search_insert_neighborhood;   search_swap_neighborhood;
       if (bEscape )
       {  bestMove = the most disproving move
       }
       else
       {   bestMove=the best non-tabu move, unless there is a tabu move leads to a solution superior to all solution so far
           visited, in which case this tabu move will be chosen
       }
       if no move is available
       {  Clear tabu;  ts_tenure = max( DECREASE * ts_tenure, ts_tenure_min );
          Remove the first solution from the list g_goodSolutions, use it as a starting solution
       }
       else
       {   perform bestMove and get new solution solNext
           if there is no solution with the same obj. values as solNext stored in list g_goodSolutions
           {   store solNext into the list g_goodSolutions
               sort the list g_goodSolutions by ascending obj. value
           }
```

```
 // now check if the new solution has been visited before
if (solNext found in g_mapSolutionHistory)
{   update in solNext in g_mapSolutionHistory: update lastVisitedIteration, repetitions++
    if ( repetitions > REP ) && ( ! OftenVisited ))
    {  OftenVisited = true;
       chaotic ++;
       if ( chaotic > CHAOS )
       {    // now set the flag of escape
            bEscape = true;   bClearHistory = true;   nRemainingEscapeSteps = min( nMovingAvg/3, 2 );
            chaotic = 0;   use the current best solution as the starting point
    } }
    // check what how many iterations between this visit and last visit for this solution
    nCycleLength = it – lastVisitedIteration;
    if (nCycleLength < CYCLE_MAX )
    {   // adjust moving average and tenure
        nMovingAvg = 0.1 * nCycleLength + 0.9 * nMovingAvg;
        ts_tenure = min(  INCREASE * ts_tenure, ts_tenure_max);
     }
    nStepsSinceLastSizeChange = 0;
}
// adjust ts_tenure
if ( nStepsSinceLastSizeChange > nMovingAvg )
{  ts_tenure = max( DECREASE * ts_tenure, ts_tenure_min );  nStepsSinceLastSizeChange = 0;
}
// put the new solution into history if it has not been found in the history
if ( not found )
{   put solNext into g_mapSolutionHistory;
}
//Now set tabu tenure, either (a) Route-Position-Flight Tabu //attribute, or (b) Sub-route First Flight Tabu
   attribute which is decided before solving the problem
if (  (a) attributes )
{ if ( bestMove is an insert move )
   {for the leading flight to be inserted,  set the tenure for the triplets (flights being inserted, old position, old
       route);
     key = ( nPositionNo * 100 + nRouteNo) * 100 + nFlightNo;
     g_tabu_list_map[ key ] = current_iterationNo + ts_tenure;
   }
  else if ( bestMove is a swap move )
```

```
            { for the leading flight to be swapped in both sub-route, set the tenure for the triplets (flights being swaped,
                old position,old route);
              key = (nPositionNo * 100 + nRouteNo) * 100 + nFlightNo;
              g_tabu_list_map[ key ] = current_iterationNo + ts_tenure;
          }   }
        else // it is  (b) tabu attributes
        {
            if (bestMove is an insert move ) { set the tenure for flights being inserted}
            else if ( bestMove is a swap insert )  { set the tenure for the flights being swapped  }
} } } }
```

Appendix B

Multicommodity Network Flow Model for AGP

For all ground arcs associated with one station, define $a_1 < a_2$ if a_1 is temporally earlier than a_2 and define $a_1 <= a_2$ if a_1 is temporally no later than a_2.

Define the sets and parameters as follows:

Sets

S: stations; each nodes is associated with one station

T: types of arcs (flight, grounding)

Parameter

station(n)) : station associated with node *n*

time(n) : timestamp associated with node *n*

ori(a): origin node of arc *a*

des(a): destination node of arc *a*

o_a : origin station of arc *a*, or *station(ori(a))*

d_a : terminal station of arc *a, station(des(a))*

t_a : type of arc *a.*

In the definitions below, suffix *t* is ignored when referring to arcs since they are of the same suffix *t* (belong to the same commodity, or aircraft).

Define a relationship ***precede*** between two flight arcs *a* and *b*. Flight arc *a precede b* if $d_a = o_b$ and time(d_a) <= time(o_b)that is, (1) *a* terminates at the same station as *b* origins, and (2) the timestamp of *a*'s terminal node is no later than that of *b*'s origin node. Figure B.1 shows an example of "arcs *a* ***precedes*** *b* ."

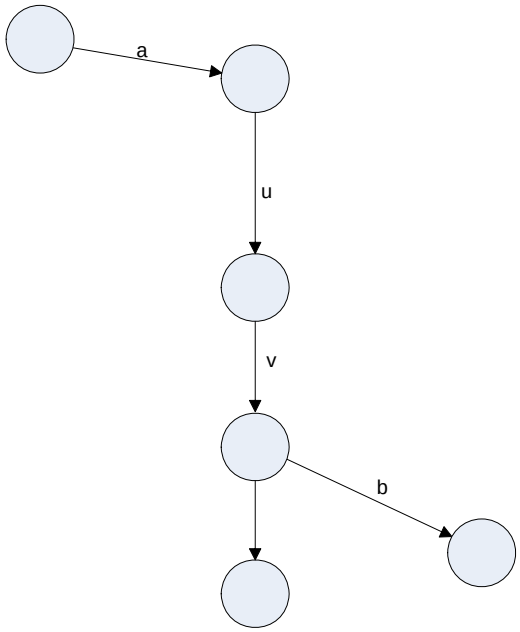

Figure B.1: Flight Arc *a* Precedes Flight Arc *b*

Then, for flight arcs *a* and *b* stated above, define set

Arcs(*a, b*) ={ ground arc $l \mid d_l = d_a$, *time(des(a) <= time(ori(l)), time(des(l))<= time(ori(b))*} ∪ {*a, b*}.

That is, set *Arcs (a, b)* is the set of all grounding arcs connecting arc *a* and *b* union with {*a, b*}. In the example shown in Figure B.1, *Arcs(a, b)* = {*u, v, a, b* }.

For flight f_1 and f_2, if f_1 follows f_2 in the original route, define set

$P(f_1, f_2)$ = {(*a, b*)| *a* is one of flight arcs of f_1, *b* is one of flight arcs of f_2, *a* precedes *b*}.

That is, $P(f_1, f_2)$ is the set of all pair of flight arcs associated with f_1 and f_2 respectively which sustains the time sequence between f_1 and f_2.

Based on the above definition, in the new routes, the necessary and sufficient condition of that flight f_1 is followed by f_2 is:

$$\exists t \in T, (a,b) \in P(f_{1,}f_2), \text{ s.t. } x_{gt} = 1, \forall g \in Arcs(a,b).$$

For the example shown in Figure B.1, flight f_1 is followed by f_2 if and only if: there is a route t satisfying

$$x_{ut} = x_{vt} = x_{at} = x_{bt} = 1.$$

Given the definitions above, the mathematical model may be stated as follows:

Indices and sets:

n: node

a: arcs between two nodes

t: aircraft (commodity)

f: flights

A: set of arcs

F: set of flights

D_t: demand nodes for aircraft t

T: aircraft

$I(n)$: sets of arcs entering node n

$O(n)$: sets of arcs leaving node n

$Z(f)$: sets of flight arcs associated with flight f

Parameters:

C_a: cost of arc at, for all t

s_t: supply node for aircraft t

d_n: demand at node n

β_f: cancellation cost of flight f

K: penalty coefficient for swaps

P: penalty exponent for swaps

$$N(f_1, f_2) = \begin{cases} 1, & \text{if flight } f_1 \text{ follows } f_2 \text{ in the original route} \\ 0, & \text{otherwise} \end{cases}$$

Variables:

x_{at}: flow on arc a (binary) for aircraft t

y_f: indicates if flight f is to be cancelled (binary)

$$r(f_1, f_2) = \begin{cases} 0, & \text{if flight } f_1 \text{ follows } f_2 \\ 1, & \text{otherwise} \end{cases}$$

$$m(f_1, f_2, t) = \begin{cases} 1, & \text{if flight } f_1 \text{ follows } f_2 \text{ in route } t \\ 0, & \text{otherwise} \end{cases}$$

$$y(a,b,t) = \begin{cases} 1, & \text{if all arcs belonging to } Arcs(a,b) \text{ in route } t \text{ are of value 1} \\ 0, & \text{otherwise} \end{cases}$$

Minimize

$$\sum_{t\in T}\sum_{a\in A} C_a x_{at} + \sum_{f\in F} \beta_f y_f + K\left[\sum_{f_1,f_2,st.N(f_1,f_2)=1} r(f_1,f_2)\right]^P$$

Subject to:

$$\sum_{t\in T}\sum_{a\in I(n)} x_{at} = d_n, \forall n \in D \quad (1)$$

$$\sum_{a\in O(s_t)} x_{at} = 1 \forall t \in T \quad (2)$$

$$\sum_{a\in I(n)} x_{at} = \sum_{a\in O(n)} x_{at}, \forall n \in N\setminus(D\bigcup S), t \in T \quad (3)$$

$$\sum_{t\in T}\sum_{a\in Z(f)} x_{at} + y_f = 1, \forall f \in F \quad (4)$$

$$r(f_1,f_2) + \sum_{t\in T} m(f_1,f_2,t) = 1, \forall f_1,f_2 \quad \text{with} \quad N(f_1,f_2)=1 \quad (5)$$

$$m(f_1,f_2,t) \le \sum_{(a,b)\in P(f_1,f_2)} y(a,b,t), \forall f_1,f_2 \quad \text{with} \quad N(f_1,f_2)=1, \forall t \in T \quad (6')$$

$$y(a,b,t) \le \sum_{g\in Arcs(a,b)} x_{gt}, \forall g \in Arcs\,(a,b) \quad (7')$$

$$y_f \in \{0,1\}, \forall f \in F \quad (8)$$

$$x_{at} \in \{0,1,...\}, \forall a \in A, t \in T \quad (9)$$

$$r(f_1,f_2) \in \{0,1\}, \forall N(f_1,f_2)=1 \quad (10)$$

$$m(f_1,f_2,t) \in \{0,1\}, \forall N(f_1,f_2)=1, t \in T \quad (11)$$

$$y(a,b,t) \in \{0,1\}, \forall (a,b) \in P(f_1,f_2) \ \ with \ N(f_1,f_2)=1, \ t \in T. \quad (12)$$

The objective function is the sum of cancellation cost, delay cost and swap penalty, which is to be minimized. Constraint (1) and (2) state the flow entering and leaving the network at the beginning and end of the recovery period, respectively. The flow balance at the intermediate nodes is maintained by constraint (3). Flight coverage is ensured by constraint (4). The value of $r(f_1, f_2)$ is decided by constraint (5), which is 0 if and only if flight f_1 follows f_2 in some route. Constraint (6') and (7') mean that the value

of $m(f_1, f_2, t)$ is 1 only if in some route t flight f_1 follows f_2. Constraints (8), (9), (10), (11) and (12) are the binary variable constraints.

This model is a multicommodity network flow model with side constraints. Therefore, it is NP-hard.

The size of the model is decided by the following parameters:

Number of aircraft (a) Number of stations (s)

Number of flights (f) Number of delay options (d).

The numbers of the constraints are as follows:

Constraints (1)*: a* Constraints (5): *O(f)*

Constraints (2): *a* Constraints (6'): *O(f · a)*

Constraints (3): *O(a · f · d)* Constraints (7'): *(d · d · f · a)*

Constraints (4): *f.*

The numbers of binary variables are as follows:

x_{at}: *O(a · f · d)*

y_f: *O(f)*

$r(f_1, f_2)$: *O(f)*

$m(f_1, f_2, t)$: O(*f · a)*

y(a, b, t): O(d · d · f · a).

Appendix C

Comparison of Computational Results from Three Algorithms for AGP

Table C.1 presents the overall objective value and time used of the results of the 60 randomly selected problems for three methods: AGP-TS, HM1 and HM2.

aircraft grounded	AGP-TS	HM1	HM2	AGP-TS Found At Time (s)	Time HM1 Used (s)	HM2: Found at Time (s)	HM2 improved HM1?	HM2: improvement percentage
18	11331	9570	9261	15	294	1	Y	3.23%
4	10631	9570	9261	16	293	2	Y	3.23%
21	9134	9570	9261	22	299	1	Y	3.23%
19	10398	9464	9280	19	306	1	Y	1.95%
6	11448	11361	11361	20	305	0	N	0.00%
22	15130	11591	11504	22	294	1	Y	0.75%
7	13449	12753	12595	7	312	7	Y	1.24%
1	14414	12570	12570	25	294	0	N	0.00%
25	9178	9570	9261	24	292	1	Y	3.23%
17	17910	12490	12490	11	309	0	N	0.00%
5	11268	9464	9280	5	299	1	Y	1.95%
10	15011	13888	13888	0	294	0	N	0.00%
23	9278	9570	9261	19	296	2	Y	3.23%
7	13449	12753	12595	6	310	6	Y	1.24%

Table C.1: Individual Result Comparison of Three Methods for AGP: Overall Objective Value and Time
(Page 1 of 3)

aircraft grounded	AGP-TS	HM1	HM2	AGP-TS Found At Time (s)	Time HM1 Used (s)	HM2: Found at Time (s)	HM2 improved HM1?	HM2: improvement percentage
16	15266	13451	13451	17	294	0	N	0.00%
2 18	28060	25148	25028	0	300	2	Y	0.48%
5 19	23373	24159	23406	26	297	1	Y	3.12%
6 20	27148	25711	25560	17	299	17	Y	0.59%
15 25	26440	22651	22651	31	293	0	N	0.00%
4 16	25173	24694	24208	23	294	2	Y	1.97%
1 22	26435	25761	24910	29	303	42	Y	3.30%
12 23	23965	26837	24758	28	295	64	Y	7.75%
18 19	19838	21967	19710	24	291	52	Y	10.28%
1 17	33384	26580	26580	14	295	0	N	0.00%
3 9	25133	25000	23540	28	294	46	Y	5.84%
7 14	28453	22965	22523	6	294	24	Y	1.93%
2 16	30433	29681	29681	27	295	0	N	0.00%
4 20	26429	23798	23504	18	319	2	Y	1.23%
10 16	30265	28708	28528	14	293	2	Y	0.63%
16 24	29864	23770	23770	41	299	0	N	0.00%
17 19 21	34474	37327	35234	25	295	2	Y	5.61%
4 10 13	40270	39214	38728	18	303	2	Y	1.24%
11 20 24	43585	36488	36488	34	309	0	N	0.00%
6 9 20	46050	41138	40691	62	297	53	Y	1.09%
16 20 24	45520	39333	38775	38	294	24	Y	1.42%
7 9 22	39328	40255	38959	32	294	58	Y	3.22%
16 25 26	37325	38384	36290	38	297	61	Y	5.46%

Table C.1: Individual Result Comparison of Three Methods for AGP: Overall Objective Value and Time
(Page 2 of 3)

aircraft grounded	AGP-TS	HM1	HM2	AGP-TS Found At Time (s)	Time HM1 Used (s)	HM2: Found at Time (s)	HM2 improved HM1?	HM2: improvement percentage
11 13 14	35901	37221	36690	25	293	2	Y	1.43%
7 12 15	42755	43664	41416	52	307	54	Y	5.15%
1 5 10	41940	39460	38020	29	299	2	Y	3.65%
13 21 23	37801	36081	34748	21	294	59	Y	3.69%
11 19 25	39944	31393	31039	29	291	1	Y	1.13%
11 14 16	38658	37211	37024	22	292	2	Y	0.50%
6 9 17	46381	43418	42984	63	293	0	Y	1.00%
13 16 23	39205	39380	38520	46	291	4	Y	2.18%
1 2 22 24	55185	53524	51096	53	294	3	Y	4.54%
0 13 14 20	54800	52120	51858	25	298	2	Y	0.50%
3 12 14 17	58713	56030	52351	42	292	3	Y	6.57%
1 11 12 16	55666	55731	55464	82	296	1	Y	0.48%
6 8 23 26	71559	55162	52823	30	292	47	Y	4.24%
6 10 13 18	53891	52284	51660	38	304	70	Y	1.19%
11 13 23 25	47280	48880	48671	30	295	2	Y	0.43%
2 12 14 26	54983	53562	51969	33	296	3	Y	2.97%
1 9 16 24	53284	51503	51503	27	299	0	N	0.00%
6 14 17 25	52741	55677	53504	40	323	47	Y	3.90%
0 2 6 11	60354	53563	53563	14	294	0	N	0.00%
4 12 13 16	56845	52716	51865	19	293	1	Y	1.61%
13 21 24 25	47835	44905	44463	36	293	30	Y	0.98%
1 3 8 10	60145	54094	52990	44	295	1	Y	2.04%
3 15 19 26	54779	55360	53001	35	295	60	Y	4.26%
average	33981	32002	31268	27	298	15		3.23%
sum							48	

Table C.1: Individual Result Comparison of Three Methods for AGP: Overall Objective Value and Time
(Page 3 of 3)

Table C.2 presents the statistics of the results of the 60 randomly selected problems for three methods: AGP TS, HM1 and HM2.

	AGP-TS				HM1				HM2			
aircraft grounded	**cancel**	**cancel Cost**	**delay cost**	**swap**	**cancel**	**cancel Cost**	**delay Cost**	**swap**	**cancel**	**cancel Cost**	**delay Cost**	**swap**
18	4	10620	0	8	2	2970	5000	12	2	2970	5180	10
4	4	10620	0	1	2	2970	5000	12	2	2970	5180	10
21	2	5010	3580	7	2	2970	5000	12	2	2970	5180	10
19	4	8820	1400	4	2	3240	4880	11	2	3240	5140	9
6	4	9090	2180	4	4	7050	3600	8	4	7050	3600	8
22	7	15030	0	3	4	6900	3980	8	4	6900	4060	7
7	5	11505	1900	2	6	11775	720	5	6	11775	720	3
1	6	14370	0	2	4	8190	3480	9	4	8190	3480	9
25	4	8100	900	4	2	2970	5000	12	2	2970	5180	10
17	7	17190	620	3	4	7110	4480	9	4	7110	4480	9
5	4	8610	2480	4	2	3240	4880	11	2	3240	5140	9
10	6	15000	0	1	4	11550	2160	4	4	11550	2160	4
23	4	8100	900	5	2	2970	5000	12	2	2970	5180	10
7	5	11505	1900	2	6	11775	720	5	6	11775	720	3
16	7	15255	0	1	4	6840	5500	10	4	6840	5500	10
2 18	10	27000	960	3	8	17010	5960	14	8	17010	6140	13
5 19	10	21855	1240	5	8	13695	7620	16	8	13695	8600	10
6 20	11	24690	2180	5	10	21060	3540	10	10	21060	3600	9
15 25	10	21900	4440	3	8	18420	3120	10	8	18420	3120	10

Table C.2: Individual Result Comparison of Three Methods for AGP: Property Statistics
(Page 1 of 3)

	AGP-TS				HM1				HM2			
aircraft grounded	**cancel**	**cancel Cost**	**delay cost**	**swap**	**cancel**	**cancel Cost**	**delay cost**	**swap**	**cancel**	**cancel Cost**	**delay cost**	**swap**
4 16	11	23835	1060	5	8	13890	7960	16	8	13890	8140	14
1 22	12	25455	880	3	8	15090	7460	17	8	15210	8100	12
12 23	10	23865	0	3	6	11580	9880	22	6	11700	10880	14
18 19	8	17880	1680	5	4	6210	9880	23	4	6330	10880	15
1 17	13	32460	880	2	10	22560	3120	9	10	22560	3120	9
3 9	12	24075	880	4	6	10740	9360	21	6	10860	10180	15
7 14	11	25875	2400	4	8	14745	5720	15	8	14745	5900	13
2 16	13	29295	960	4	10	23070	5500	10	10	23070	5500	10
4 20	12	26385	0	2	8	16980	4940	13	8	16980	5180	11
10 16	13	30165	0	3	8	18870	7660	14	8	18870	7780	13
16 24	13	29820	0	2	12	22410	960	6	12	22410	960	6
17 19 21	12	30870	3060	7	10	20970	10480	23	10	20970	11420	16
4 10 13	16	40170	0	3	14	31950	4420	16	14	31950	4600	14
11 20 24	18	43485	0	3	17	36210	0	5	17	36210	0	5
6 9 20	17	41070	2480	15	16	33780	5180	14	16	35820	3760	10
16 20 24	20	45420	0	3	16	31935	5240	14	16	31935	5240	12
7 9 22	17	36270	2780	5	14	26175	9180	21	14	26295	9820	16
16 25 26	16	32805	3620	9	14	25260	8680	20	14	28830	5860	12
11 13 14	15	34290	900	8	13	29010	5000	17	13	29010	5180	15
7 12 15	16	36375	5480	9	14	28185	9080	24	14	28305	9900	17
1 5 10	16	39060	2480	6	12	27060	7500	21	12	27060	8460	15
13 21 23	14	32790	4300	8	12	25410	6660	19	12	27450	5420	13
11 19 25	15	37500	1900	7	11	25035	4480	13	11	25035	4660	11
11 14 16	15	35820	960	13	13	26040	7960	17	13	26040	8140	16

Table C.2: Individual Result Comparison of Three Methods for AGP: Property Statistics (Page 2 of 3)

	AGP-TS				HM1				HM2			
aircraft grounded	**cancel**	**cancel Cost**	**delay cost**	**swap**	**cancel**	**cancel Cost**	**delay cost**	**swap**	**cancel**	**cancel Cost**	**delay cost**	**swap**
6 9 17	18	43530	2140	8	16	37140	4400	13	16	37140	4500	11
13 16 23	17	37905	900	6	14	29880	5900	18	14	29880	6140	15
1 2 22 24	25	51885	1700	12	21	39945	7180	24	21	39945	7940	17
0 13 14 20	23	52800	1600	6	20	45180	4440	15	20	45180	4500	14
3 12 14 17	22	54435	2100	14	16	35160	12160	28	16	35160	13180	19
1 11 12 16	23	53895	1060	8	21	45540	6980	17	21	45540	7080	16
6 8 23 26	30	69735	1280	7	19	41385	8400	22	19	41505	9140	14
6 10 13 18	21	51900	1280	8	20	45120	4320	16	20	47160	2900	12
11 13 23 25	19	45480	900	9	17	40200	5080	18	17	40200	5260	17
2 12 14 26	23	53745	960	5	19	38745	9440	22	19	38745	10380	16
1 9 16 24	25	52680	60	7	22	43125	6200	14	22	43125	6200	14
6 14 17 25	23	51990	40	8	18	41100	8700	23	18	41220	9440	16
0 2 6 11	25	57210	2600	7	21	48165	3520	13	21	48165	3520	13
4 12 13 16	23	55245	0	12	19	40785	7920	19	19	40785	8580	15
13 21 24 25	20	46935	500	6	20	39405	3000	15	20	39405	3180	13
1 3 8 10	24	56925	2320	9	18	42510	7140	20	18	42510	7980	15
3 15 19 26	23	47295	6940	7	18	36810	11040	26	18	36930	12060	19
average	13.9	32048	1463	5.7	11.1	23268	5863	15.1	11.1	23447	6057	12.1

Table C.2: Individual Result Comparison of Three Methods for AGP: Property Statistics (Page 3 of 3)

Appendix D

Result of HM+TS for RSC Problem: Additional Experiment

Table D.1 presents the result of the experiments of starting RSC-TS from the solution yielded by HM, where all MOG violations were prohibited. Note the superior results for problems 6, 7 and 10 when compared to the best results in Table 4.6.

Problem Instance	Objective Function Values	Time Used at Finding Best Solution
1	2,504	116
2	7,381	123
3	13,480	146
4	88,657	122
5	10	68
6	**1,430**	114
7	**3,854**	164
8	69,786	79
9	1,154	35
10	**2,298**	35
11	19,921	35
12	654	13
13	1,220	13
14	16,008	16

Table D.1: Result of HM+TS: Start RSC-TS from the Solutions Yielded by HM Prohibiting All MOG Violations

Bibliography

Andersson, T. (2001) *The flight perturbation problem – operational aircraft rescheduling*, Lic. Thesis, Linköping University

Andersson, T., P. Värbrand (2004) The flight perturbation problem, *Transportation Planning and Technology*, 227(2), 91–117

Andersson, T. (2006) Solving the flight perturbation problem with meta heuristics, *Journal of Heuristics* 12(1-2), 37-53

Argüello, M., J. F. Bard, G. Yu (1997) A GRASP for aircraft routing in response to groundings and delays, *Journal of Combinatorial Optimization*, 5, 211-228

Argüello, M. (1997) *Framework for exact solutions and heuristics for approximate solutions to airlines' irregular operations control aircraft routing problem*, Ph.D. Dissertation, Department of Mechanical Engineering, University of Texas, Austin

Bailey, T. G. (2007) Principal and general manager, Analytic Designs LLC, personal communication, November 7

Bard, J. F., G. Yu, M. F. Argüello (2001) Optimizing aircraft routings in response to groundings and delays, *IIE Transactions*, 33, 931-947

Barnes, J. W., J. Chambers (1995) Solving the Job Shop Scheduling Problem using Tabu Search, *IIE Transactions*, 27, 257-263

Barnes, J. W., M. Laguna (1993) Solve the multiple-machine weighted flow time problem using tabu search*, IIE Transactions*, 25, 121-128

Barnes, J. W., V. Wiley, J. Moore, and D. Ryer (2004) Solving the aerial fleet refueling problem using group theoretic tabu search, *Mathematical and Computer Modeling*, 6-8

Battiti, R., G. Techiolli (1994) The reactive tabu search, *ORSA Journal on Computing*, 6(2), 126-140

Carlton, W., J. W. Barnes (1996) Solving the Traveling Salesman Problem with Time Windows Using Tabu Search, *IIE Transactions*, 28, 617-629

Crino, J.R., J.T. Moore, J.W. Barnes, and W.P. Nanry (2004) Solving the Theater Distribution Vehicle Routing and Scheduling Problem Using Group Theoretic Tabu Search, *Mathematical and Computer Modelling*, 39(6-8), 599-616

Filar J. A., P. Manyem, K. White (2001) How airlines and airports recover from schedule perturbations: a survey, *Annals of Operations Research*, 108, 315-333

Garey, M.R., Johnson, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York

Glover, F. and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers, Norwell, MA

Glover, F., Laguna, M., and Martí, R. (2000) Fundamentals of Scatter Search and Path Relinking, *Control and Cybernetics*, 39(3), 653-684

González-Velarde, J. L. and M. Laguna (2002) Tabu search with simple ejection chains for coloring graphs, *Annals of Operations Research*, 117, 165-174.

Harwig, J., J. W. Barnes, J. T. Moore, An adaptive tabu search approach for 2-dimensional orthogonal packing problems (2006) *Military Operations Research*, 11(2), 5-26

Jarrah, A. I. Z., G. Yu, N. Krishnamurthy, A. Rakshit (1993) A decision support framework for airline flight cancellations and delays, *Transportation Science*, 27(3), 266-280

Kinney, G., J. W. Barnes, and B. Colletti (2007) A Reactive Tabu Search algorithm with variable clustering for the Unicost Set Covering Problem, *International Journal of Operational Research*, 2(2), 156-172

Kohl, N., A. Larsen, J. Larsen, A. Ross, S. Tiourine S (2004) Airline disruption management—perspectives, experiences and outlook, *Technical Report* 2004–16, Department of Informatics and Mathematical Modelling, Technical University of Denmark

Korycinski, D., M.M. Crawford, J.W. Barnes, and J. Ghosh (2003) Adaptive feature selection for hyperspectral data analysis using a binary hierarchical classifier and tabu search, *Proc. 2003 International Geoscience and Remote Sensing Symposium*, Toulouse, France, 297-299

Laguna, M., J.W. Barnes, F. W. Glover (1991) Tabu search methods for a single machine scheduling problem, *Journal of Intelligent Manufacturing*, 2, 63-74

Laguna, M., J. P. Kelly, J. L. González Velarde, and F. Glover (1995) Tabu search for the multilevel generalized assignment problem, *European Journal of Operational Research*, 82, 176-189

Lettovský, L., E. L.Johnson, G. L. Nemhauser (2000) Airline crew recovery, *Transportation Science*, 34(4), 337-348

Lokketangen, A., F. Glover (1998) Solving zero-one mixed integer programming problems using tabu search, *European Journal of Operational Research*, 106, 1998, 624-658

Loeve, M., K. R. Soerensen, J. Larsen, J. Clausen (2001a) *Using heuristics to solve the dedicated aircraft recovery problem*, Technical report, Technical University of Denmark

Loeve, M., K. R. Soerensen (2001b) *Disruption Management in the Airline Industry*, M.Sc. thesis, Department of Informatics and Mathematical Modeling at the Technical University of Denmark, Lyngby, February 28, 2001

Loeve, M., Soerensen, K. R., Larsen, J., and Clausen, J. (2002) Disruption management for an airline - rescheduling of aircraft, *Proceedings of the Applications of Evolutionary Computing on Evoworkshops 2002: Evocop, Evoiasp, EvoSTIM/EvoPLAN*, April 03-04

Luo S., G. Yu (1997) On the airline schedule perturbation problem caused by the ground delay problem, *Transportation Science*, 31(4), 298-311

Nanry, W. P., J. W. Barnes (2000) Solving the pickup and delivery problem with time windows using reactive tabu search, *Transportation Research* Part B (34), 107-121

Pachon, J. (2007a) Navitaire Inc., personal communication, January 8

Pachon, J. (2007b) Navitaire Inc., personal communication, June 22

Rosenberger J. M., E. L. Johnson, G. L. Nemhauser (2003) Rerouting Aircraft for Airline Recovery, *Transportation Science*, 37(4), 408-421

Scrich, C. R., V. A. Armentano and M. Laguna (2004) Tardiness minimization on a flexible job shop: a tabu search approach, *Journal of Intelligent Manufacturing*, 15(1), 103-115

Thengvall, B. G., J. F. Bard, G. Yu (2000) Balancing user preferences for aircraft schedule recovery during irregular operations, *IIE Transactions*, 32, 181-193

Thengvall, B. G., G. Yu, J. F. Bard (2001) Multiple fleet aircraft schedule recovery following hub closures, *Transportation Research*, Part A 35, 289-308

Van der Bruggen, L. J. J.; J. K. Lenstra, P. C. Schuur (1993) Variable-Depth Search for the Single-Vehicle Pickup and Delivery Problem with Time Windows, *Transportation Science*, 27(3), 298-311

Vranas, P. B., B. J. Bertsimas, A. R. Odoni (1994) The multi-airport ground-holding problem in air traffic control, *Operations Research*, 42(2), 249-262

Wei, G., G. Yu, M. Song (1997) Optimization model and algorithm for crew management during airline irregular operations, *Journal of Combinatorial Optimization*, 1, 305-321

Yan, S., C. Lin (1997), Airline Scheduling for the Temporary Closure of Airports, Transportation Science, 31(1), Feb. 1997, 72-82

Yan, S., D. H. Yang (1996) A decision support framework for handling schedule perturbation, *Transportation Research*, Part B, 30(6) 405-419

Yu, G., M. Argüello, G. Song, S. M. McCowan (2003) A. White, A new era for crew recovery at continental airlines, *Interface*s, INFORMS, Jan.-Feb. 2003, 5-22

Yu, G., X. Qi (2004) *Disruption management, framework, models and applications*, World Scientific

Vita

Mei Yang was born in Qinghai, China on December 13, 1973, the fourth daughter of Kezheng Yang and Rongzhi Li. In September 1990 upon graduation from Hainan No. 1 High School, Qinghai, China, she entered Tsinghua University in Beijing, China. She received the degree of Bachelor of Engineering in 1995 and Master of Engineering in 1998, both in Computer Sciences. In the following year she worked for Beijing Telecom in Beijing, China. In September 1999 she entered Graduate School of The University of Texas at Austin. In December 2000 she received the degree of Masters of Science in Engineering in Operations Research and Industrial Engineering. After graduation she was employed by the Center for Space Research of The University of Texas at Austin as a Research Engineer/Scientist Associate and then by Emerson Process Management in Austin, Texas as a Software Engineer. In September 2003 she began doctoral studies in the Operations Research and Industrial Engineering Program at The University of Texas at Austin.

Permanent address: 3517 North Hills Drive, Apt. W201, Austin, Texas, 78731

This dissertation was typed by the author.